

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỆ THÔNG THÔNG TIN



BÁO CÁO ĐO ÁN MÔN HỌC
KHAI THÁC DỮ LIỆU

Đề tài:

DỰ ĐOÁN KHẢ NĂNG HỦY ĐẶT PHÒNG TỪ DỮ LIỆU
HOẠT ĐỘNG CỦA CÁC KHÁCH SẠN TỪ NĂM 2015 ĐẾN
2017

Giảng viên hướng dẫn: **ThS. Hà Lê Hoài Trung**

Lớp: **IS252.M21**

Sinh viên thực hiện:

Đặng Vũ Phương Uyên - 19520345

Nguyễn Ngọc Thiện - 19522264

Nguyễn Thị Thu Phương - 19522066

TP.HCM, ngày 20 tháng 05 năm 2022

MỤC LỤC

CHƯƠNG 1: NHẬN DIỆN BÀI TOÁN KHAI THÁC DỮ LIỆU	1
1. Tổng quan về dữ liệu	1
1.1. Lý do chọn đề tài	1
1.2. Giới thiệu nguồn dữ liệu	1
2. Mô tả thuộc tính	2
3. Phát biểu bài toán	9
4. Công cụ khai thác dữ liệu	9
5. Thư viện kèm theo	9
CHƯƠNG 2: TRỰC QUAN HÓA DỮ LIỆU TRƯỚC KHI TIỀN XỬ LÝ	11
1. Thông kê loại hình khách sạn nào thu nhiều khách hàng nhất	11
2. Thông kê tỷ lệ nhận phòng và hủy phòng của hai loại hình khách sạn	12
3. Thông kê tỷ lệ số lượt đặt phòng thành công hoặc bị hủy	13
4. Tỷ lệ các trạng thái đăng ký khi hủy bỏ phòng thành công	14
5. Thông kê hệ số tương quan giữa giá trị hủy đặt phòng và các thuộc tính còn lại ..	15
6. Thông kê tỷ lệ phòng bị hủy	17
7. Thông kê doanh thu mỗi tháng trong tập dữ liệu	18
8. Thông kê tỷ lệ hủy đặt phòng với từng loại hình đặt cọc	21
9. 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	22
10. Thông kê thời gian và tổng doanh thu của khách sạn	23
11. Thông kê tên các quốc gia và số lượng du khách thấp hơn 1000\$	23
CHƯƠNG 3: TIỀN XỬ LÝ DỮ LIỆU	25
1. Mô tả dữ liệu	25
2. Làm sạch dữ liệu	26
2.1. Nhận diện các dữ liệu bị nhiễu	26
2.2. Xử lý dữ liệu bị nhiễu	27
2.3. Nhận diện các dữ liệu bị thiếu	30
2.4. Xử lý ý nghĩa của dữ liệu	33
2.5. Xử lý dữ liệu bị thiếu	34
2.6. Xử lý các giá trị NaN	35

2.7. Kiểm tra thuộc tính của dữ liệu	36
2.8. Xử lý các dữ liệu bị trùng lặp	38
3. Tích hợp dữ liệu	38
4. Biến đổi dữ liệu và thu giảm dữ liệu	40
4.1. Thêm các thuộc tính.	40
4.2. Thu giảm dữ liệu.....	41
4.3. Chuẩn hóa dữ liệu.....	42
5. Rời rạc hóa dữ liệu	43
6. Sampling	47
7. Biểu diễn dữ liệu	48
8. Thu giảm số chiều dữ liệu.....	49
CHƯƠNG 4: ÚNG DỤNG GIẢI THUẬT PHÂN LỚP VÀO TẬP DỮ LIỆU.....	51
1. K- Fold Cross Validation	51
2. Baseline Model	52
2.1. Khái niệm Baseline Model	52
2.2. Ưu điểm	52
2.3. Giải thuật Baseline.....	53
3. Decision Tree Model	54
3.1. Khái niệm.....	54
3.2. Ưu điểm	54
3.3. Decision Tree ID3 Model	55
3.4. Decision Tree Cart Model	57
4. Naïve Bayes Model	59
4.1. Khái niệm Naïve Bayes Model.....	59
4.2. Ưu điểm	59
4.3. Giải thuật Naïve Bayes	60
5. Random Forest Model.....	61
5.1. Khái niệm Random Forest Model	61
5.2. Ưu điểm	61
5.3. Giải thuật Random Forest.....	62
6. Logistic Regression Model.....	63
6.1. Khái niệm Logistic Regression Model	63

6.2.	Ưu điểm	64
6.3.	Giải thuật Logistic Regression	64
7.	K - Nearest Neighbor Model	66
7.1.	Khái niệm K - Nearest Neighbor Model	66
7.2.	Ưu điểm	66
7.3.	Giải thuật K - Nearest Neighbor	66
8.	Neural Network Model.....	68
8.1.	Khái niệm Neural Network Model	68
8.2.	Ưu điểm	68
8.3.	Giải thuật Neural Network	69
9.	Bagging Classifier Model.....	70
9.1.	Khái niệm Bagging Classifier Model	70
9.2.	Ưu điểm	70
9.3.	Giải thuật Bagging Classifier	70
CHƯƠNG 5: PHÂN TÍCH ĐÁNH GIÁ CÁC THUẬT TOÁN VÀ DỰ BÁO		72
1.	Ma trận nhầm lẫn.....	72
2.	Đánh giá tỷ lệ TPR (True positive rate) trên tỷ lệ FPR (False positive rate)..	72
3.	Đánh giá về thời gian chạy thuật toán	73
4.	Đánh giá về độ chính xác của thuật toán	73
5.	Đánh giá thuật toán.....	74
5.1.	So sánh thời gian chạy	74
5.2.	So sánh độ chính xác của tập huấn luyện	75
5.3.	So sánh độ chính xác của tập thử nghiệm	76
5.4.	So sánh các thuật toán	77
6.	Đánh giá làm việc	79
BẢNG PHÂN CHIA CÔNG VIỆC.....		80
TÀI LIỆU THAM KHẢO.....		81

CHƯƠNG 1: NHẬN DIỆN BÀI TOÁN KHAI THÁC DỮ LIỆU

1. Tổng quan về dữ liệu.

1.1. Lý do chọn đề tài.

Trong những năm qua với sự phát triển của ngành du lịch thì sự đóng góp không nhỏ vào thành công chung của ngành phải kể đến trước hết là lĩnh vực kinh doanh khách sạn. Trong đó bộ phận lễ tân khách sạn nói chung và hoạt động đặt phòng nói riêng là nhân tố chủ chốt giúp thúc đẩy hoạt động kinh doanh của khách sạn.

Đối với khách hàng, quá trình đặt phòng làm tăng cơ hội về một thỏa thuận cho chỗ ở, khách lưu trú sẽ yên tâm khi đến nơi. Đối một khách sạn, dữ liệu đặt phòng có thể giúp quản lý tìm xu hướng và đưa ra dự báo về kinh doanh và quản lý tốt hơn trải nghiệm của khách trong suốt mùa kinh doanh bình thường cũng như mùa cao điểm.

Nhận thấy được tầm quan trọng của quá trình đặt phòng trong lĩnh vực khách sạn, nhóm đã quyết định chọn đề tài phân tích hoạt động đặt phòng của các khách sạn từ 2015-2017.

1.2. Giới thiệu nguồn dữ liệu

1.2.1. Tên nguồn dữ liệu.

Tên dataset: Hotel Booking Demand.

Ngày cập nhật gần nhất: 06/2021.

1.2.2. Đơn vị cung cấp.

Đây là dữ liệu được lấy từ bài báo *Hotel Booking Demand Datasets* trong tạp chí *Data in Brief* số 22 xuất bản vào tháng 02/2019 được viết bởi Nuno Antonio, Ana Almeida và Luis Nunes.

1.2.3. Mô tả số dòng, cột và thời gian thu thập dữ liệu.

Bộ dữ liệu gồm 119390 dòng dữ liệu và 36 cột thuộc tính chứa quan sát các hoạt động đặt phòng từ các city hotel và resort hotel.

Dữ liệu được thu nhập vào khoảng thời gian từ 01/07/2015 – 31/08/2017

1.2.4. Hướng chủ đề:

Tập dữ liệu được xây dựng với hướng chủ đề kinh doanh khách sạn.

Nguồn tải dataset: <https://www.kaggle.com/mojtaba142/hotel-booking>

2. Mô tả thuộc tính.

STT	Tên thuộc tính	Ý nghĩa	Kiểu dữ liệu
1	Hotel	Kiểu khách sạn. Có hai kiểu: 1. Resort Hotel. 2. City Hotel.	Nominal
2	Is_canceled	Hủy đặt phòng. Có hai giá trị: 1. Giá trị 0: không hủy. 2. Giá trị 1: hủy.	Quantitative (Asymmetric)
3	Lead_time	Số ngày từ ngày khách hàng đặt chỗ và được ghi nhận vào hệ thống quản lý của khách sạn (PSM) đến ngày khách hàng đến nhận phòng.	Quantitative (Ratio-scaled)
4	Arrival_date_time	Năm của ngày khách hàng đến.	Quantitative (Interval -scaled)
5	Arrvial_date_month	Tháng của ngày khách hàng đến với 12 giá trị từ tháng 1 đến tháng 12.	Quantitative (Interval -scaled)
6	Arrvial_date_week_number	Số thứ tự tuần trong năm của ngày khách hàng đến với giá trị từ 1 đến 53.	Quantitative (Interval -scaled)

7	Arrvial_date_of_month	Số ngày trong tháng của ngày khách hàng đến với giá trị từ 1 đến 31.	Quantitative (Interval -scaled)
8	Stays_in_weekend_nights	Số đêm cuối tuần (thứ bảy hoặc chủ nhật) mà khách có lưu trú hoặc đặt phòng tại khách sạn	Quantitative (Ratio-scaled)
9	Stays_in_week_nights	Số đêm trong tuần (từ Thứ Hai đến Thứ Sáu) mà khách đã lưu trú hoặc đặt phòng tại khách sạn (được tính bằng cách đếm số đêm trong tuần).	Quantitative (Ratio-scaled)
10	Adults	Số lượng khách hàng là người lớn	Quantitative (Ratio-scaled)
11	Children	Số lượng khách hàng là trẻ em	Quantitative (Ratio-scaled)
12	Babies	Số lượng khách hàng là em bé	Quantitative (Ratio-scaled)
13	Meal	<p>Cho biết các bữa ăn được bao gồm trong giá phòng mà khách hàng đặt, có 3 giá trị chính:</p> <ul style="list-style-type: none"> + 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 	Nominal

		<p>trưa thay vì bữa sáng - khách sạn sẽ xác nhận điều này khi khách hàng đến.</p> <p>+ FB (Full Board): Được bao gồm bữa sáng, bữa trưa và buổi tối.</p>	
14	Country	<p>Quốc tịch của khách hàng.</p>	Nominal
15	Market_segment	<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 (vd : khi một chuyến bay bị hủy, hãng hàng không có thể sẽ sắp xếp cho khách hàng chỗ nghỉ trong khách sạn và khi check- out khách hàng không phải trả tiền)</p>	Nominal

		<ul style="list-style-type: none"> + Corporate: khách hàng đến từ các công ty, doanh nghiệp + Groups: khách hàng đến từ các hội nhóm, tổ chức + Direct: khách hàng đặt phòng trực tiếp tại khách sạn + Undefined: không xác định 	
16	Distribution_channel	<p>Để tiếp cận với lượng lớn khách thuê phòng tiềm năng, các khách sạn đều liên kết bán phòng qua nhiều kênh phân phối phòng khách sạn. Có ba loại:</p> <ul style="list-style-type: none"> + 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. 	Nominal
17	Is_repeated_guest	<p>Cho biết lượt đặt chỗ có phải từ khách hàng đã lưu trú nhiều lần tại khách sạn hay là lần đầu đến</p> <p>Có hai giá trị:</p>	Binary

		<ol style="list-style-type: none"> 1. 0: khách hàng tới lần đầu. 2. 1: khách hàng cũ. 	
18	Previous_cancellations	Số lượng lượt đặt phòng đã bị khách hàng hủy trước lượt đặt phòng hiện tại.	Quantitative (Ratio -scaled)
19	Previous_booking_not_cancelled	Số lượng lượt nhận phòng thành công của khách hàng trước lượt đặt phòng hiện tại.	Quantitative (Ratio -scaled)
20	Reserved_room_type	Mã loại phòng được đại diện bằng các chữ cái.	Ordinal
21	Assigned_room_type	Mã loại phòng được chỉ định cho lượt đặt phòng. Đôi khi loại phòng được chỉ định khác với loại phòng đã đặt vì lý do hoạt động của khách sạn (khách sạn nhận số lượng đặt phòng nhiều hơn số lượng phòng thực tế) hoặc do yêu cầu của khách hàng.	Ordinal
22	Booking_changes	Số lượng sửa đổi đối với lượt đặt phòng kể từ thời điểm đặt phòng được nhập trên PMS cho đến thời điểm nhận phòng hoặc lượt đặt phòng bị hủy bỏ	Quantitative (Ratio -scaled)
23	Deposit_type	Thể hiện hình thức đặt cọc trước: + No Deposit: không có tiền đặt cọc trước + Non-Refund: một khoản đặt cọc đã được thực hiện bằng giá trị của tổng chi phí lưu trú.	Nominal

		+ Refundable: một khoản đặt cọc đã được thực hiện với giá trị dưới tổng chi phí lưu trú.	
24	Agent	Thể hiện ID của đại lý du lịch đã đặt chỗ	Quantitative (Interval -scaled)
25	Company	ID của công ty / tổ chức đã thực hiện đặt phòng hoặc chịu trách nhiệm thanh toán đặt phòng	Quantitative (Interval -scaled)
26	Days_in_waiting_list	Số ngày lượt đặt chỗ ở trong danh sách chờ trước khi nó được xác nhận với khách hàng	Quantitative (Ratio -scaled)
27	Customer_type	Thể hiện loại khách hàng: + Group: Khi khách hàng đặt phòng theo một nhóm. + Transient: Khách tạm trú là những khách lưu trú qua đêm không có ý định ở lại lâu dài. + Transient party: Khách tạm trú theo nhóm lưu trú qua đêm không có ý định ở lại lâu dài.	Nominal
28	Adr	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.	Quantitative (Ratio – scaled)

29	Required_car_parking_spaces	Số chỗ đậu xe ô tô theo yêu cầu của khách hàng.	Quantitative (Ratio – scaled)
30	Total_of_special_requests	Số lượng yêu cầu đặc biệt của khách hàng (ví dụ: có giường đôi hoặc ở tầng cao)	Quantitative (Ratio – scaled)
31	Reservation_status	<p>Thể hiện tình trạng đặt phòng:</p> <ul style="list-style-type: none"> + 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. 	Nominal
32	Reservation_status_date	Ngày mà trạng thái cuối cùng của lượt đặt phòng được cập nhật. Thuộc tính này có thể được sử dụng cùng với Reservation_status để biết khi nào đặt phòng bị hủy hoặc khi nào khách hàng đã trả phòng khách sạn.	Datetime
33	Name	Họ tên của khách hàng	Nominal
34	Email	Email của khách hàng	Nominal
35	Phone_number	Số điện thoại khách hàng	Nominal
36	Credit_card	Số thẻ tín dụng của khách hàng	Nominal

3. Phát biểu bài toán.

Dựa vào những thuộc tính đã có sẵn trong tập dữ liệu, dự đoán mối quan hệ giữa khả năng hủy đặt phòng khách sạn khoanh vùng đối tượng phục vụ tốt hơn và đề ra các chính sách phù hợp khi hủy đơn tăng cao.

4. Công cụ khai thác dữ liệu.

- Công cụ khai thác:
 - o Google Colab
 - o Jupyter Notebook.
- Phiên bản Python: 3.8.

5. Thư viện kèm theo.

```
import numpy as np
import pandas as pd
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
import re
import missingno as msno
import folium
import plotly.express as px
import plotly.graph_objects as go
import sort_dataframeby_monthorweek as sd
import pickle
from folium.plugins import HeatMap
from plotly.subplots import make_subplots
from scipy.stats.mstats import winsorize
from scipy import stats
import warnings
from sklearn.model_selection import train_test_split
```

```
from sklearn.dummy import DummyClassifier
from sklearn import tree
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn import svm
from sklearn.svm import SVC
from sklearn.ensemble import BaggingClassifier
from datetime import datetime, timedelta
from sklearn.model_selection import ShuffleSplit, cross_val_score, train_test_split
, KFold, GridSearchCV
import warnings
warnings.filterwarnings("ignore")
```

CHƯƠNG 2: TRỰC QUAN HÓA DỮ LIỆU TRƯỚC KHI TIỀN XỬ LÝ

- ❖ Dữ liệu ban đầu gồm 35 cột và 119390 dòng dữ liệu:

```
● df.info()
df <class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 36 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   hotel            119390 non-null   object 
 1   is_canceled      119390 non-null   int64  
 2   lead_time         119390 non-null   int64  
 3   arrival_date_year 119390 non-null   int64  
 4   arrival_date_month 119390 non-null   object 
 5   arrival_date_week_number 119390 non-null   int64  
 6   arrival_date_day_of_month 119390 non-null   int64  
 7   stays_in_weekend_nights 119390 non-null   int64  
 8   stays_in_week_nights 119390 non-null   int64  
 9   adults            119390 non-null   int64  
 10  children          119386 non-null   float64
 11  babies             119390 non-null   int64  
 12  meal               119390 non-null   object 
 13  country            118902 non-null   object 
 14  market_segment     119390 non-null   object 
 15  distribution_channel 119390 non-null   object 
 16  is_repeated_guest  119390 non-null   int64  
 17  previous_cancellations 119390 non-null   int64  
 18  previous_bookings_not_canceled 119390 non-null   int64  
 19  reserved_room_type  119390 non-null   object 
 20  assigned_room_type  119390 non-null   object 
 21  booking_changes    119390 non-null   int64  
 22  deposit_type       119390 non-null   object 
 23  agent              103650 non-null   float64
 24  company            6797 non-null    float64
 25  days_in_waiting_list 119390 non-null   int64  
 26  customer_type      119390 non-null   object 
 27  adr                119390 non-null   float64
 28  required_car_parking_spaces 119390 non-null   int64  
 29  total_of_special_requests 119390 non-null   int64  
 30  reservation_status 119390 non-null   object 
 31  reservation_status_date 119390 non-null   object 
 32  name               119390 non-null   object 
 33  email              119390 non-null   object 
 34  phone-number       119390 non-null   object 
 35  credit_card        119390 non-null   object 
dtypes: float64(4), int64(16), object(16)
memory usage: 32.8+ MB
```

Hình 2.1: Thông tin tập dữ liệu ban đầu

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

- Lấy ra các giá trị có trong thuộc tính hotel của tập dữ liệu.

```
● df['hotel'].value_counts()
City Hotel      79330
Resort Hotel    40060
Name: hotel, dtype: int64
```

Hình 2.2: Lấy giá trị trong cột hotel.

- Đếm tần suất xuất hiện của các giá trị trong tập dữ liệu và sắp thêm theo thứ tự tăng dần của tần suất.

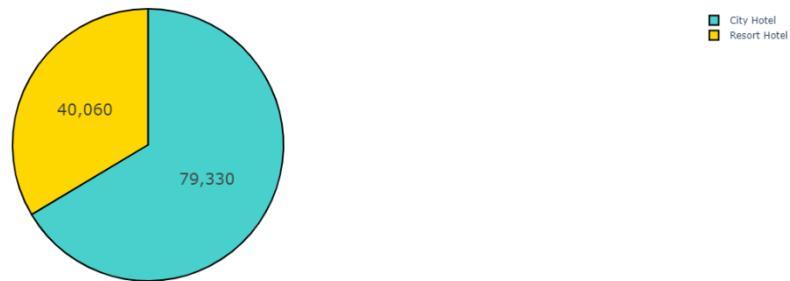
```
● labels = ["Resort Hotel", "City Hotel"]
hotel = df.hotel.value_counts().sort_values()
hotel
df Resort Hotel    40060
      City Hotel     79330
Name: hotel, dtype: int64
```

Hình 2.3: Lấy ra số lượng dòng dữ liệu theo giá trị Resort Hotel và City Hotel.

- Dùng Dashboard có sẵn trong Python (Plotly) để vẽ biểu đồ thống kê thể hiện loại hình khách sạn được ưa chuộng.

```
[18] colors = ['gold', 'mediumturquoise']

fig = go.Figure(data=[go.Pie(labels=labels,
                               values=hotel)])
fig.update_traces(hoverinfo='label+percent', textinfo='value', textfont_size=20,
                    marker=dict(colors=colors, line=dict(color='#000000', width=2)))
fig.show()
```



Hình 2.4: Kết quả thống kê.

- Kết luận:
 - Khách sạn thành phố (City Hotel) được đặt phòng nhiều nhất.
 - Lượt hủy đặt phòng của “City Hotel” cao hơn so với “Resort Hotel”, chiếm gần 50% lượng đăng ký thuê phòng trước nên sẽ mang đến nhiều ảnh hưởng tiêu cực đến City Hotel.

2. Thống kê tỷ lệ nhận phòng và hủy phòng của hai loại hình khách sạn

- Lần lượt tạo hai biến “canceled” và “not_canceled” để lưu giá trị nhận phòng hoặc hủy đặt phòng và đến số lượng của từng trường hợp trong từng kiểu khách sạn.

```
[19] canceled = df[df['isCanceled']==1][['hotel']]
canceled = canceled.hotel.value_counts().sort_values()

not_canceled = df[df['isCanceled']==0][['hotel']]
not_canceled = not_canceled.hotel.value_counts().sort_values()
```

Hình 2.5: Tạo hai biến canceled và not_canceled.

- Dùng Dashboard để vẽ biểu đồ thống kê sự phân bố dữ liệu của hai nhóm nhận phòng và hủy đặt phòng

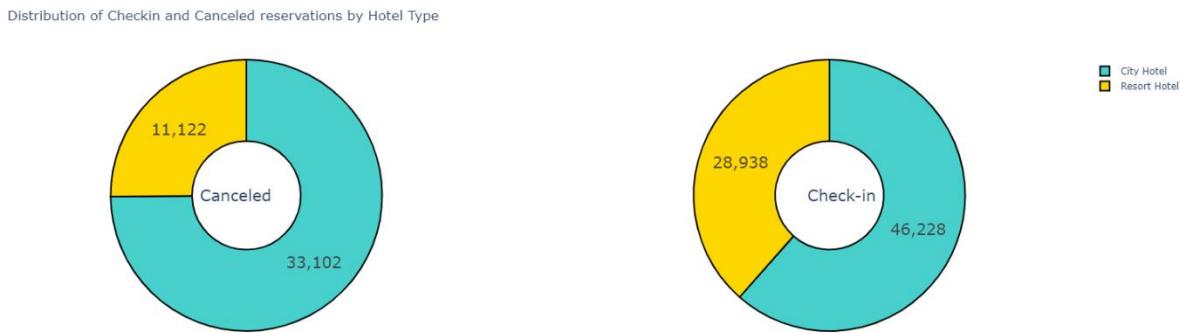
```
[22] fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])
    fig.add_trace(go.Pie(labels=labels, values=canceled, name="Canceled"),
                  1, 1)
    fig.add_trace(go.Pie(labels=labels, values=not_canceled, name="Check-in"),
                  1, 2)

    fig.update_traces(hole=.4, hoverinfo="label+percent+name", textinfo='value', textfont_size=20,
                      marker=dict(colors=colors, line=dict(color='#000000', width=2)))

    fig.update_layout(
        title_text="Distribution of Checkin and Canceled reservations by Hotel Type",
        annotations=[dict(text='Canceled', x=0.18, y=0.5, font_size=20, showarrow=False),
                     dict(text='Check-in', x=0.82, y=0.5, font_size=20, showarrow=False)])
fig.show()
```

Hình 2.6: Đoạn mã vẽ biểu đồ tròn trong Plotly.

- Kết quả thu được:



Hình 2.7: Kết quả biểu đồ tròn thể hiện sự phân bố của dữ liệu.

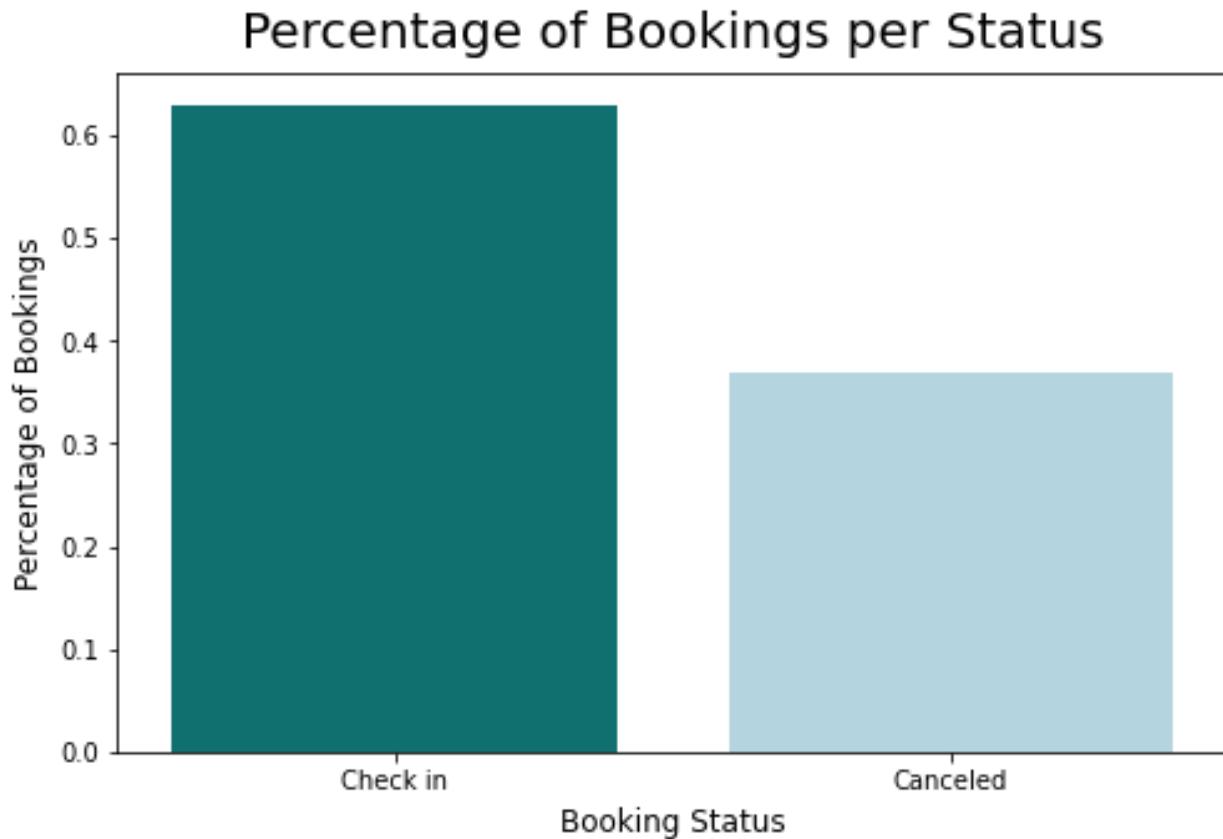
3. Thống kê tỷ lệ số lượt đặt phòng thành công hoặc bị hủy

- Sử dụng hàm barplot để xây dựng biểu đồ cột trong Seaborn.

```
▶ plt.figure(figsize=(8,5))
plt.title("Percentage of Bookings per Status", fontsize = 20, pad = 10)
sns.barplot(x=df['is_canceled'].unique(), y=df['is_canceled'].value_counts(normalize=True), palette=['teal', 'lightblue'])
plt.xlabel("Booking Status", fontsize = 12, labelpad = 5)
plt.ylabel("Percentage of Bookings", fontsize = 12, labelpad = 5)
plt.xticks(ticks=[0, 1], labels=['Check in', 'Canceled']);
```

Hình 2.8: Đoạn mã vẽ biểu đồ cột.

- Kết quả thu được:



Hình 2.9: Biểu đồ thể hiện phần trăm nhận phòng và hủy đặt phòng tại khách sạn.

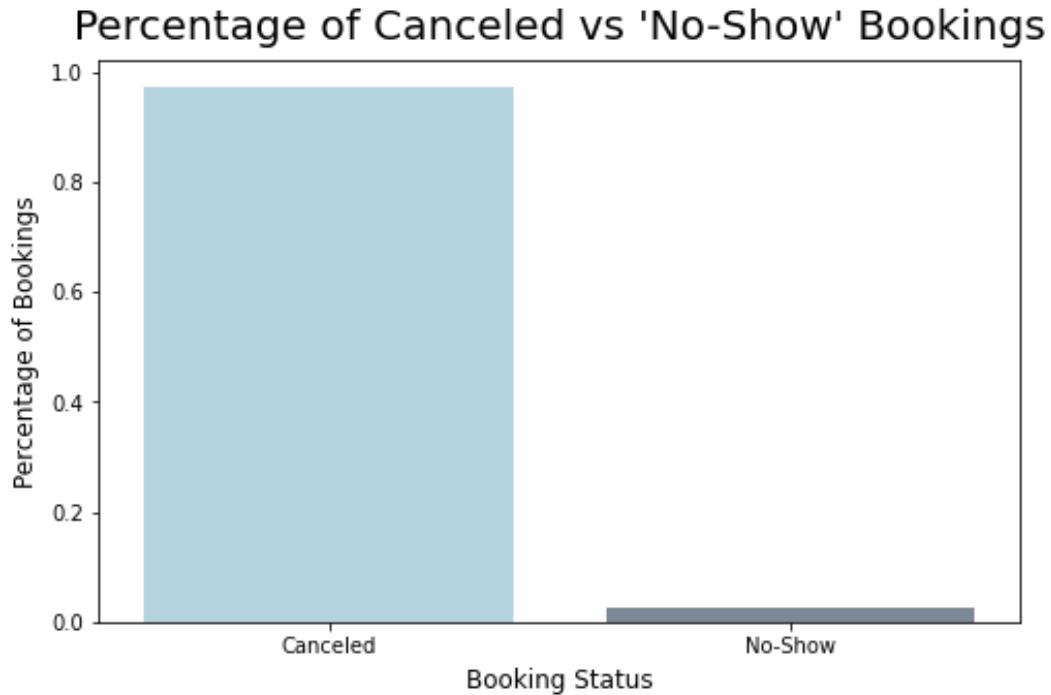
- Kết luận: Hơn một phần ba lượt đặt phòng bị hủy bỏ.

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

- Sử dụng hàm barplot trong Seaborn để xây dựng đồ thị dạng cột thống kê tỷ lệ các trạng thái cuối cùng của từng dòng dữ liệu khi có tình trạng là hủy đặt phòng (is_canceled = 1).

```
[24] plt.figure(figsize=(8,5))
plt.title("Percentage of Canceled vs 'No-Show' Bookings", fontsize = 20, pad = 10)
sns.barplot(x=df[df['is_canceled']==1]['reservation_status'].unique(),
            y=df[df['is_canceled']==1]['reservation_status'].value_counts(normalize=True),
            palette=['lightblue', 'lightslategrey'])
plt.xlabel("Booking Status", fontsize = 12, labelpad = 5)
plt.ylabel("Percentage of Bookings", fontsize = 12, labelpad = 5);
```

Hình 2.10: Đoạn mã vẽ biểu đồ thống kê.



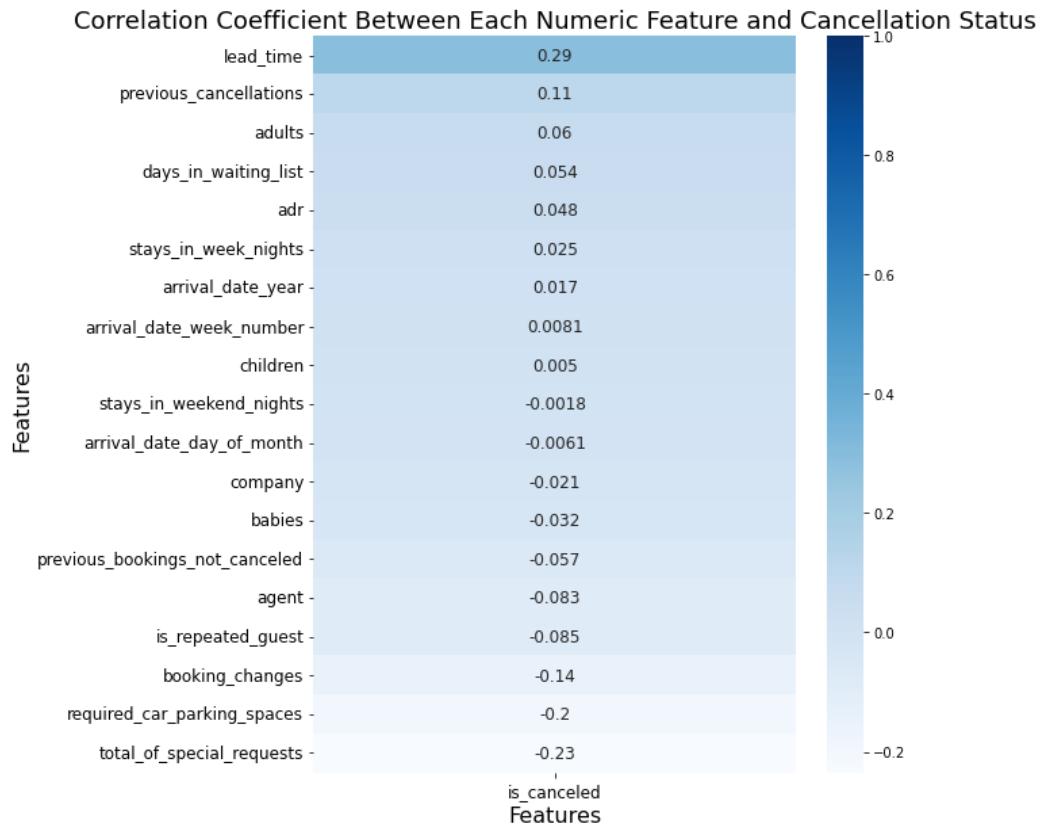
Hình 2.11: Biểu đồ thống kê tỷ lệ giữa trạng thái hủy phòng và trạng thái không có thông tin khi giao dịch bị hủy bỏ.

- Kết luận:
 - 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, ngược lại còn gây ra sai số cho thuật toán.
- 5. **Thông kê 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.

```
[25] fig = plt.figure(figsize=(8,10))
    ax = sns.heatmap(df.corr()[['is_canceled']].sort_values('is_canceled', ascending=False),
                      annot = True, annot_kws = {"size":12}, cmap='Blues')
    ax.set_title('Correlation Coefficient Between Each Numeric Feature and Cancellation Status', fontsize=18)
    ax.set_xlabel('Features', fontsize = 16)
    ax.set_ylabel('Features', fontsize = 16)
    ax.tick_params(axis = "both", labelsize = 12);
    y_min, y_max = ax.get_ylim()
    ax.set_ylimits(top=y_max+1);
```

Hình 2.12: Đoạn mã sử dụng hàm heatmap.

- Kết quả thu được:



Hình 2.13: Biểu đồ nhiệt độ thể hiện sự tương quan trong tập thuộc tính “is_canceled”.

- Trong đó, khi 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.
- Kết luận:
 - Lead_time có mối quan hệ tương quan cao nhất với việc có hủy đặt phòng hay không? 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 và có nhiều thời gian để phát sinh các tình huống không lường trước được kế hoạch du lịch.

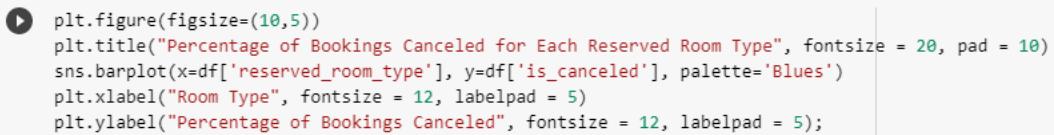


Hình 2.14: Biểu đồ tính tỷ lệ các yêu cầu đặc biệt và biến độc lập.

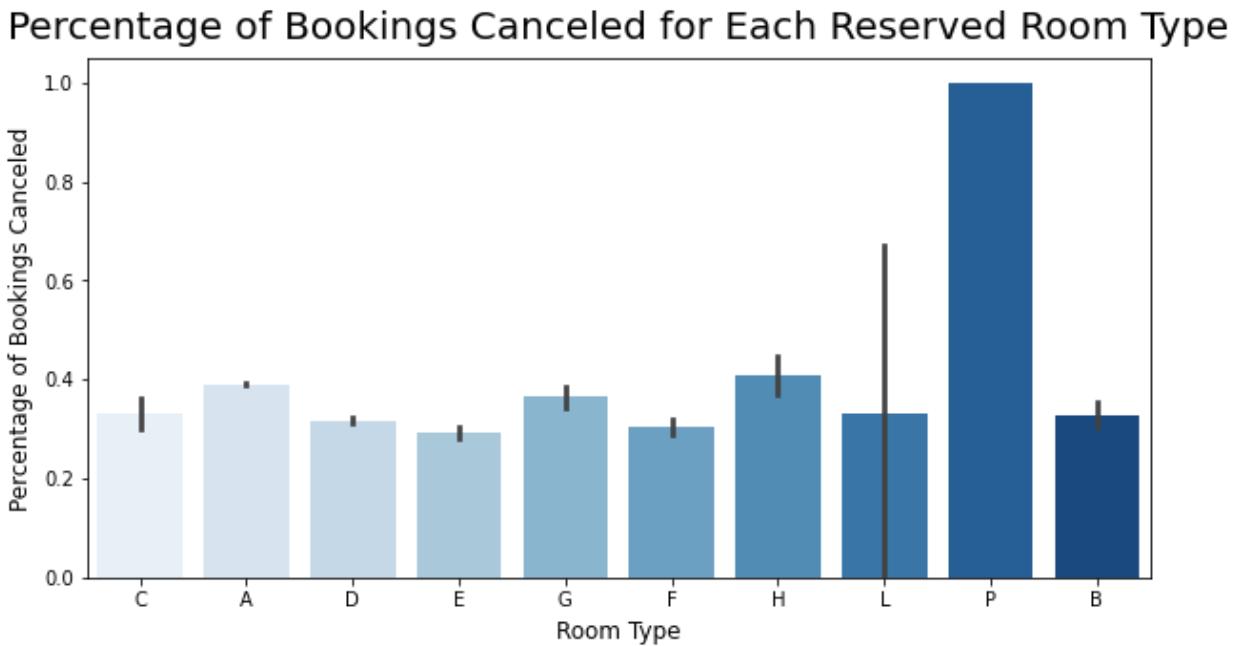
- Kết luận:
 - Required_car_parking_spaces có mối tương quan mạnh với mục tiêu hủy đặt phòng. Khi số lượng yêu cầu chỗ đậu xe tăng lên, khả năng đặt chỗ sẽ bị giảm xuống. Tương tự số lượng yêu cầu đặc biệt, sẽ có ý nghĩa rằng khách hàng càng gắn bó với khách sạn thì khả năng hủy đặt phòng càng thấp.
 - Các dữ liệu chẵn có giá trị ảnh hưởng đến kết quả “Canceled” không đáng kể, có khả năng thuộc tính này sẽ không giúp được trong quá trình khai thác. Ngược lại, lại còn gây ra sai sót cho thuật toán có thể xóa nó đi.

6. Thông kê tỷ lệ phòng bị hủy.

- Sử dụng hàm Barplot trong Seaborn để vẽ đồ thị biểu hiện dưới dạng cột.



- Kết quả thu được:



Hình 2.15: Biểu diễn tỷ lệ hủy đặt phòng của các phòng đã đăng ký.

- Kết luận:

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

7. Thông 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.

```
[33] data_resort = df[(df['hotel'] == 'Resort Hotel') & (df['is_canceled'] == 0)]
     data_city = df[(df['hotel'] == 'City Hotel') & (df['is_canceled'] == 0)]
```

Hình 2.16: Tạo hai Dataframe để lưu giá trị nhận phòng thành công của hai loại khách sạn.

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

```
[ ] resort_hotel = data_resort.groupby(['arrival_date_month'])['adr'].mean().reset_index()
```

	arrival_date_month	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 2.17: Giá trị trung bình theo từng tháng của khách sạn nghỉ dưỡng.

```
▶ city_hotel=data_city.groupby(['arrival_date_month'])['adr'].mean().reset_index()
```

	arrival_date_month	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 2.18: Giá trị trung bình theo từng giá của khách sạn trong thành phố,

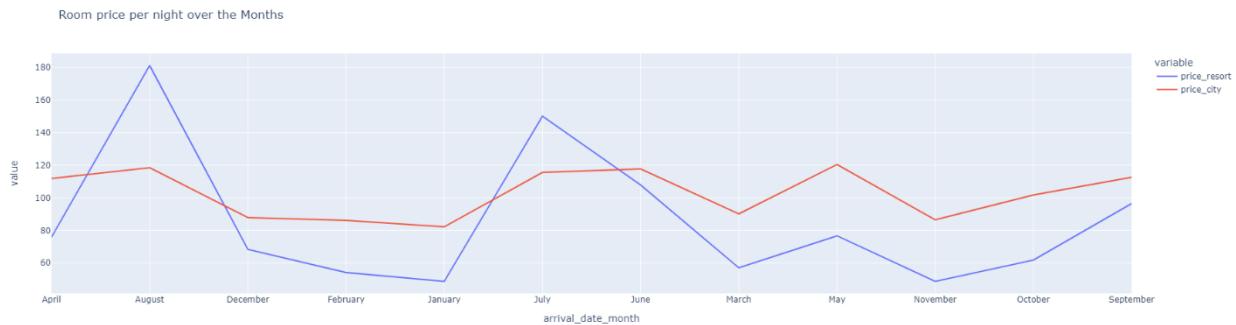
- Ghép hai Dataframe vừa tạo lại thành một tập Dataframe mới chứa doanh thu trung bình của từng tháng của hai loại hình khách sạn.

```
final_hotel = city_hotel
final_hotel["price_resort"] = resort_hotel["adr"]
final_hotel.rename(index=str,columns={"adr":'price_city'},inplace=True)
final_hotel
```

	arrival_date_month	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 2.19: Kết quả sau khi thực hiện câu lệnh.

- Kết quả đồ thị:



Hình 2.20: So sánh doanh thu của hai loại hình bằng biểu đồ đường.

- Kết luận:
 - Tháng 8 và 9 là hai khoảng thời gian đạt tới đỉnh của Resort City.
 - Doanh thu của City Hotel tương đối ổn định theo từng tháng trong từng năm.

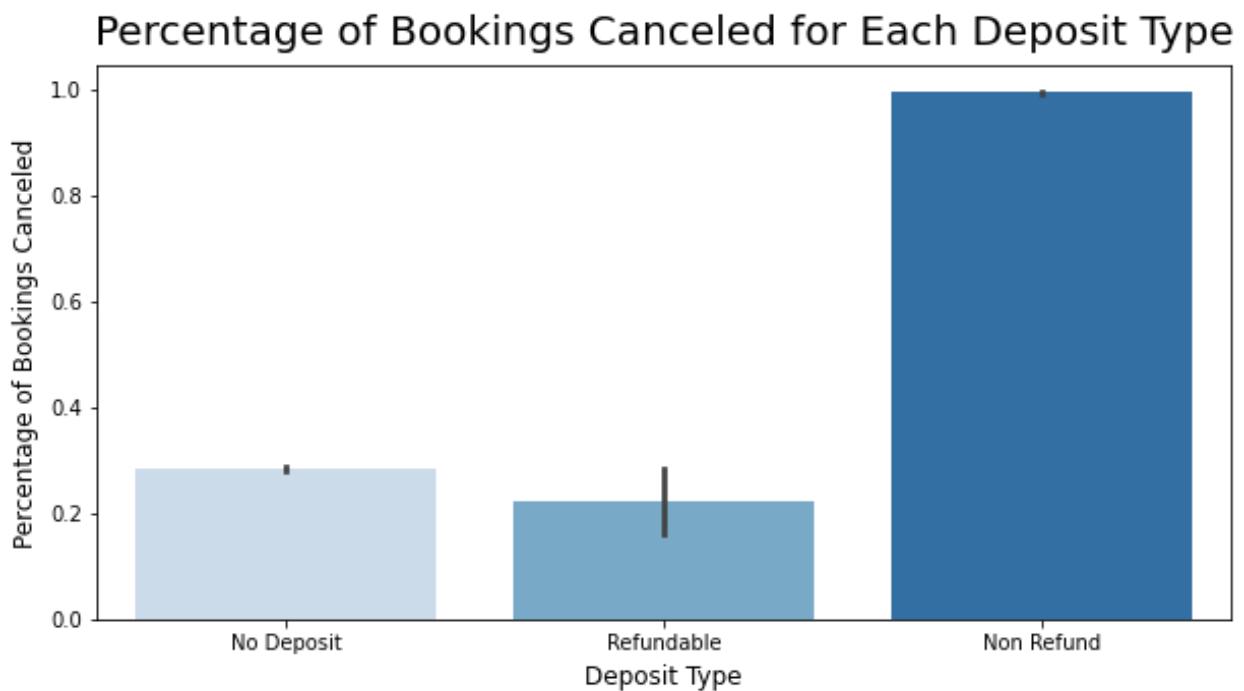
8. Thống kê tỷ lệ hủy đặt phòng với từng loại hình đặt cọc

- Sử dụng hàm Barplot trong Seaborn để tính xác suất xuất hiện của từng loại hình thanh toán.

```
plt.figure(figsize=(10,5))
plt.title("Percentage of Bookings Canceled for Each Deposit Type", fontsize = 20, pad = 10)
sns.barplot(x=df['deposit_type'], y=df['is_canceled'], palette='Blues')
plt.xlabel("Deposit Type", fontsize = 12, labelpad = 5)
plt.ylabel("Percentage of Bookings Canceled", fontsize = 12, labelpad = 5);
```

Hình 2.21: Đoạn mã chạy Barplot.

- Kết quả thu được:



Hình 2.22: Biểu đồ cột thể hiện sự chênh lệch của mỗi cách thức đăng ký phòng.

- Kết luận:

- Lượng khách hàng hủy đặt cọc không hòa lại lại có tỷ lệ không nhận phòng cao nhất.
- Tỷ lệ hủy đặt phòng ở hai giá trị còn lại xấp xỉ nhau.

9. 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.

```
[ ] labels = ["Aviation", "Complementary", "Corporate", "Groups", "Direct", "Offline TA/T0", "Online TA"]

canceled = df[df['is_canceled']==1][['market_segment']]
canceled = canceled.market_segment.value_counts().sort_values()

not_canceled = df[df['is_canceled']==0][['market_segment']]
not_canceled = not_canceled.market_segment.value_counts().sort_values()

[ ] not_canceled

Aviation          185
Complementary     646
Corporate         4383
Groups            7714
Direct             10672
Offline TA/T0     15988
Online TA          35738
Name: market_segment, dtype: int64

[ ] canceled

Undefined          2
Aviation            52
Complementary       97
Corporate           992
Direct              1934
Offline TA/T0       8311
Groups              12097
Online TA           20739
Name: market_segment, dtype: int64
```

Hình 2.23: Đoạn mã sử dụng hàm Pie để thể hiện thông tin một cách trực quan.

- Kết quả thu được:



Hình 2.24: Biểu đồ thể hiện sự phân phối của phân khúc thị trường để biết về thị trường nào thu hút nhiều khách du lịch và khả năng nhận phòng thành công của thị trường đó.

- Kết luận:
 - Hầu hết các thao tác đặt phòng và hủy phòng đều thông qua hình thức online càng phù hợp trong thời đại công nghệ hiện nay.
 - Tỷ lệ phân khúc thị trường trong việc nhận phòng và hủy phòng ở từng thị trường là tỷ lệ thuận với nhau.

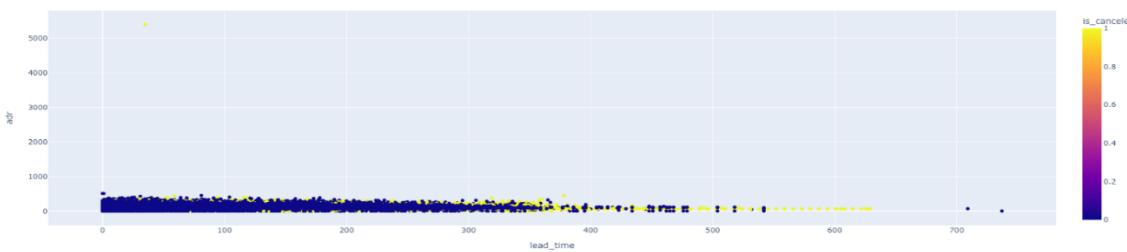
10. Thống kê thời gian và tổng doanh thu của khách sạn.

- Dùng Scatter trong Plotly để lấy độ tương quan giữa doanh thu trung bình theo ngày,

```
▶ fig = px.scatter(df, x="lead_time", y="adr", color='is_canceled')
fig.show()
```

Hình 2.25: Đoạn mã phân tích dữ liệu chi phí và thời gian dùng trong scatter.

- Kết quả thu được:



Hình 2.26: Mô tả độ tương quan với biến is_canceled với doanh thu trung bình và thời gian chờ đợi.

- Kết luận:
 - Thời gian chờ càng lâu khả năng hủy đặt phòng thành công.
 - Hầu hết doanh thu phân bố các đơn đặt phòng đều bé hơn \$1000.

11. Thống kê tên các quốc gia và số lượng du khách thấp hơn 1000\$.

- Tạo DataFrame mới tên là “guest_city” dùng để kiểm tra thông tin quốc tịch của các khách hàng nhận phòng tại khách sạn.

```
[44] guest_city = df[df['is_canceled'] == 0]['country'].value_counts().reset_index()
guest_city.columns = ['Country', 'No of guests']
guest_city
```

	Country	No of guests
0	PRT	21071
1	GBR	9676
2	FRA	8481
3	ESP	6391
4	DEU	6069
...
160	BHR	1
161	DJI	1
162	MLI	1
163	NPL	1
164	FRO	1

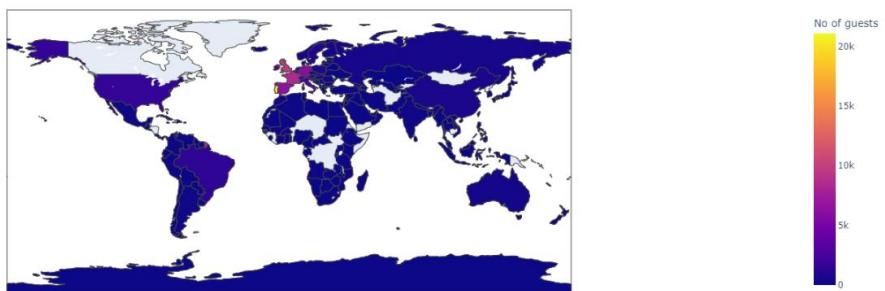
Hình 2.27: Tạo biến guest_city chứa các thông tin số lượng khách hàng đến từ mỗi nước.

- Thống kê các quốc tịch của những khách hàng nhận phòng thành công.

```
▶ basemap = folium.Map()
guests_map = px.choropleth(guest_city, locations = guest_city['Country'],
                           color = guest_city['No of guests'], hover_name = guest_city['Country'])
guests_map.show()
```

Hình 2.28: Đoạn mã thể hiện sự phân bố khách hàng đi du lịch và nhận phòng thành công của các quốc gia.

- Kết quả thu được:



Hình 2.29: Thể hiện sự phân bố lượng khách du lịch trên toàn thế giới. Trong đó, màu càng đậm lượng khách du lịch càng thừa thót.

- Kết luận:

- Hầu hết các khách hàng đến từ PRT (Portugal - Bồ Đào Nha) và một số nước châu Âu khác.
- Có thể thấy thuộc tính country có đến 178 giá trị khác nhau, có khả năng cao thuộc tính này sẽ không giúp ích được quá trình khai thác.

CHƯƠNG 3: TIỀN XỬ LÝ DỮ LIỆU

1. Mô tả dữ liệu

- Thống kê chỉ số các thuộc tính định lượng như: đếm số giá trị, giá trị lớn nhất, giá trị nhỏ nhất, trung bình, độ lệch chuẩn và tứ phân vị (thứ nhất, thứ hai và thứ ba)

	count	mean	std	min	25%	50%	75%	max
is_canceled	119390.0	0.370416	0.482918	0.00	0.00	0.000	1.0	1.0
lead_time	119390.0	104.011416	106.863097	0.00	18.00	69.000	160.0	737.0
arrival_date_year	119390.0	2016.156554	0.707476	2015.00	2016.000	2016.000	2017.0	2017.0
arrival_date_week_number	119390.0	27.165173	13.605138	1.00	16.00	28.000	38.0	53.0
arrival_date_day_of_month	119390.0	15.798241	8.780829	1.00	8.00	16.000	23.0	31.0
stays_in_weekend_nights	119390.0	0.927599	0.998613	0.00	0.00	1.000	2.0	19.0
stays_in_week_nights	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
is_repeated_guest	119390.0	0.031912	0.175767	0.00	0.00	0.000	0.0	1.0
previous_cancellations	119390.0	0.087118	0.844336	0.00	0.00	0.000	0.0	26.0
previous_bookings_not_canceled	119390.0	0.137097	1.497437	0.00	0.00	0.000	0.0	72.0
booking_changes	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
days_in_waiting_list	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
required_car_parking_spaces	119390.0	0.062518	0.245291	0.00	0.00	0.000	0.0	8.0
total_of_special_requests	119390.0	0.571363	0.792798	0.00	0.00	0.000	1.0	5.0

Hình 3.1: Thống kê thông tin chi tiết của dữ liệu định lượng.

- Thống kê thông tin của các thuộc tính định tính: đếm 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.

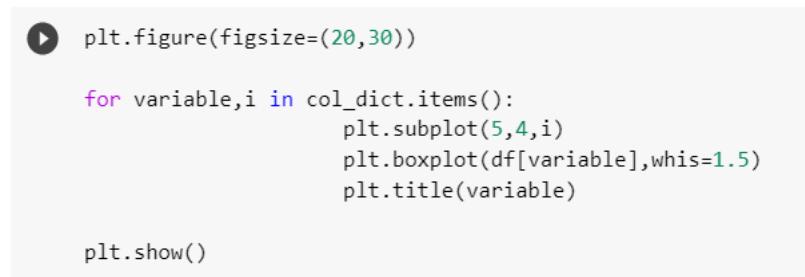
	count	unique	top	freq
hotel	119390	2	City Hotel	79330
arrival_date_month	119390	12	August	13877
meal	119390	5	BB	92310
country	118902	177	PRT	48590
market_segment	119390	8	Online TA	56477
distribution_channel	119390	5	TA/TO	97870
reserved_room_type	119390	10	A	85994
assigned_room_type	119390	12	A	74053
deposit_type	119390	3	No Deposit	104641
customer_type	119390	4	Transient	89613
reservation_status	119390	3	Check-Out	75166
reservation_status_date	119390	926	2015-10-21	1461
name	119390	81503	Michael Johnson	48
email	119390	115889	Michael.C@gmail.com	6
phone-number	119390	119390	669-792-1661	1
credit_card	119390	9000	*****4923	28

Hình 3.2: Thống kê thông tin chi tiết của dữ liệu định tính.

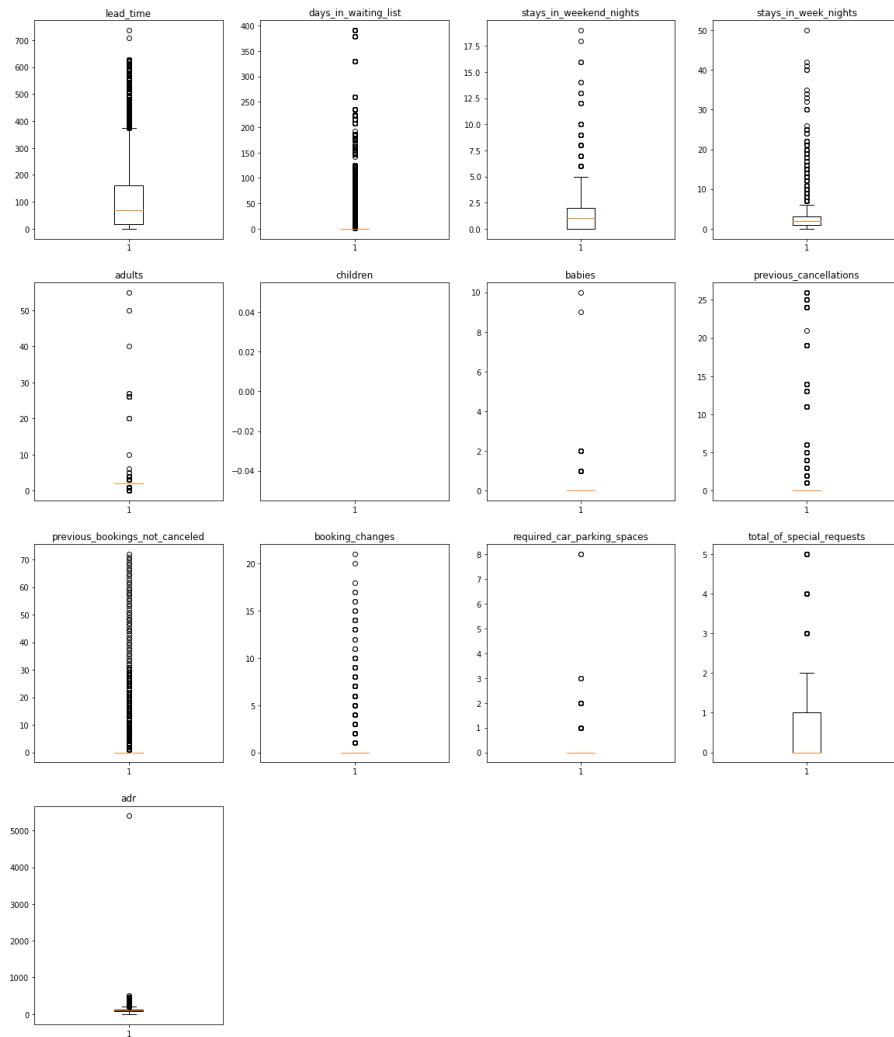
2. Làm sạch dữ liệu

2.1. Nhận diện các dữ liệu bị nhiễu.

- Vẽ biểu đồ Boxplot để xem thông tin các dữ liệu định lượng và kiểm tra các ngoại lệ (Outlier) trong từng thuộc tính số:



Hình 3.3: Biểu đồ BoxPlot cho từng thuộc tính định lượng.



Hình 3.4: Xuất biểu đồ Boxplot dưới dạng 4 cột 5 hàng.

- Thể hiện thông tin số lượng ngoại lệ và phần trăm số lượng ngoại lệ cụ thể:



```

for variable in col_dict.keys():
    q75, q25 = np.percentile(df[variable], [75, 25])
    iqr = q75 - q25

    min_val = q25 - (iqr*1.5)
    max_val = q75 + (iqr*1.5)
    print("Number of outliers and percentage of it in {} : {} and {}".format(variable,
        len((np.where((df[variable] > max_val) | (df[variable] < min_val)))[0])),len((np.where((df[variable] > max_val) | (df[variable] < min_val)))[0]))*100/len(df)))

```

Hình 3.5: Lấy các ngoại lệ nằm ngoài đoạn từ tứ phân vị thứ nhất đến tứ phân vị thứ ba.

```

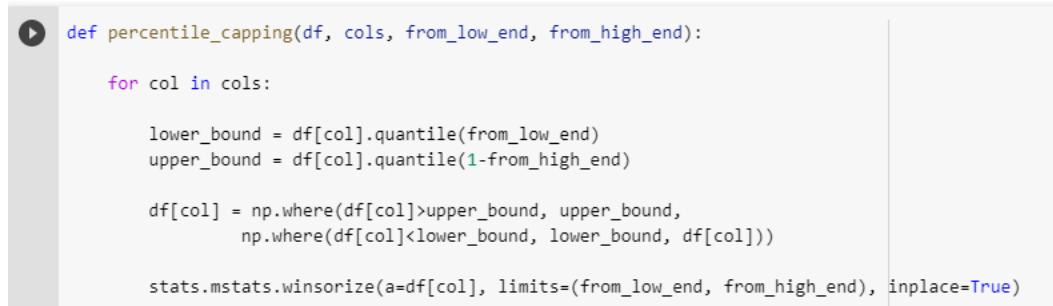
↳ Number of outliers and percentage of it in lead_time : 3005 and 2.5169612195326243
Number of outliers and percentage of it in days_in_waiting_list : 3698 and 3.097411843537985
Number of outliers and percentage of it in stays_in_weekend_nights : 265 and 0.22196163832816818
Number of outliers and percentage of it in stays_in_week_nights : 3354 and 2.8092805092553816
Number of outliers and percentage of it in adults : 29710 and 24.88483122539576
Number of outliers and percentage of it in children : 0 and 0.0
Number of outliers and percentage of it in babies : 917 and 0.768071027724265
Number of outliers and percentage of it in previous_cancellations : 6484 and 5.430940614791859
Number of outliers and percentage of it in previous_bookings_not_canceled : 3620 and 3.0320797386715803
Number of outliers and percentage of it in booking_changes : 18076 and 15.140296507245163
Number of outliers and percentage of it in required_car_parking_spaces : 7416 and 6.211575508836586
Number of outliers and percentage of it in total_of_special_requests : 2877 and 2.409749560264679
Number of outliers and percentage of it in adr : 3793 and 3.176982996900913

```

Hình 3.6: Kết quả tổng số lượng và phần trăm tỷ lệ ngoại lai của thuộc tính đó chiếm trên toàn tập dữ liệu.

2.2. Xử lý dữ liệu bị nhiễu.

- Dùng phương pháp Winsorization để loại bỏ các ngoại lệ tồn tại trong từng thuộc tính bằng cách thay thế A% giá trị đầu mảng thành giá trị gần nhất liền kề và B% giá trị cuối mảng thành giá trị gần nhất liền kề.



```

def percentile_capping(df, cols, from_low_end, from_high_end):
    for col in cols:
        lower_bound = df[col].quantile(from_low_end)
        upper_bound = df[col].quantile(1-from_high_end)

        df[col] = np.where(df[col]>upper_bound, upper_bound,
                           np.where(df[col]<lower_bound, lower_bound, df[col]))

    stats.mstats.winsorize(a=df[cols], limits=(from_low_end, from_high_end), inplace=True)

```

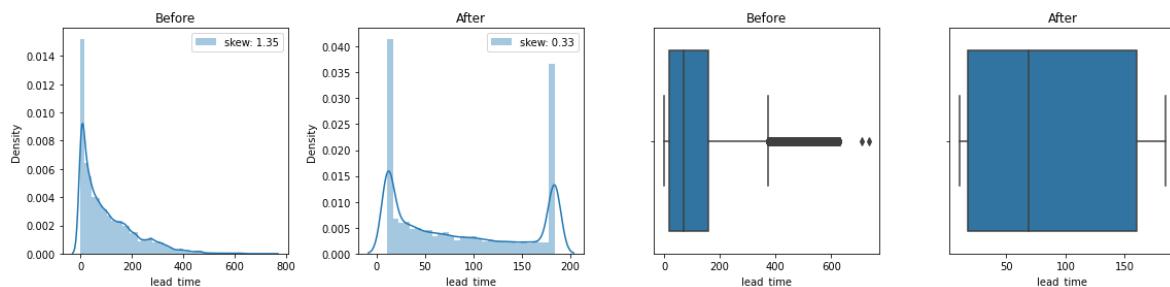
Hình 3.7: Đoạn mã để loại bỏ các giá trị Outlier.

- Lấy các cột loại bỏ các ngoại lệ:

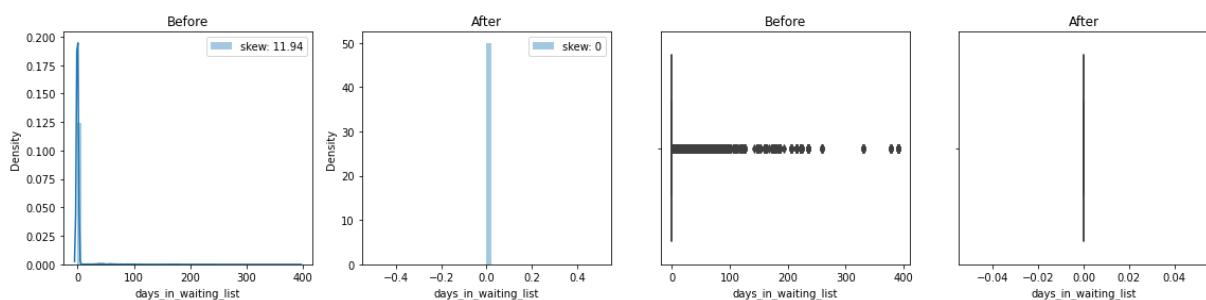
```
cols = [
    'lead_time',
    'days_in_waiting_list',
    'stays_in_weekend_nights',
    'stays_in_week_nights',
    'adults',
    'babies',
    'previous_cancellations',
    'previous_bookings_not_canceled',
    'booking_changes',
    'required_car_parking_spaces',
    'total_of_special_requests',
    'adr']
```

Hình 3.8: Các cột cần xử lý ngoại lệ.

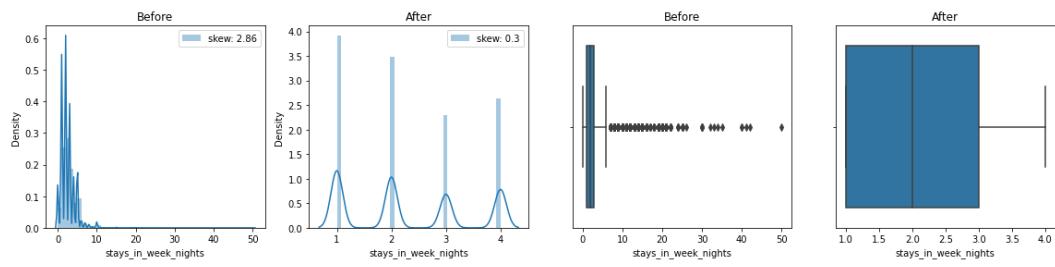
- Chạy hàm xử lý ngoại lệ và tiến hành in đồ thị để so sánh dữ liệu trước khi loại bỏ ngoại lệ và sau khi loại bỏ ngoại lệ. Trong đó hàm skew được sử dụng để tính toán độ lệch của dữ liệu hiện tại.
 - Nếu kết quả trả về là 0: đồ thị đối xứng và đạt phân phối chuẩn.
 - Nếu kết quả trả về số âm: đồ thị không đối xứng, lệch về phía bên trái,
 - Nếu kết quả trả về dương: đồ thị không đối xứng và lệch về phía bên phải.



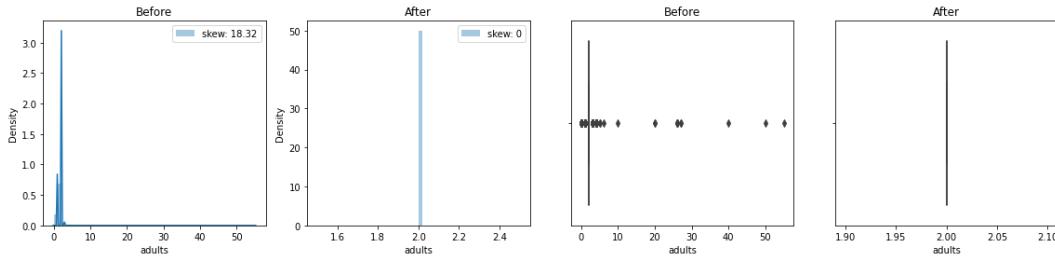
Hình 3.9: Thuộc tính lead_time



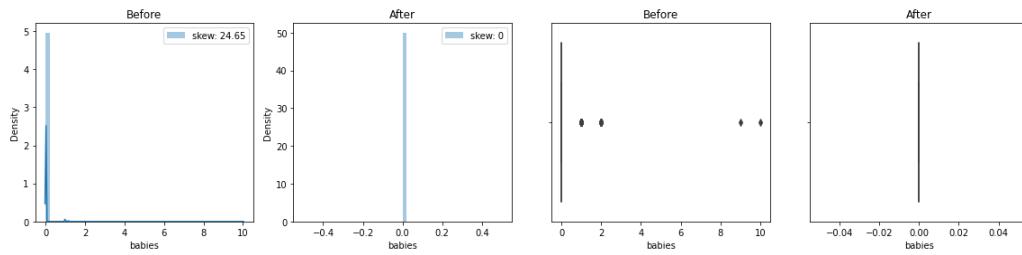
Hình 3.10: Thuộc tính days_in_waiting_list



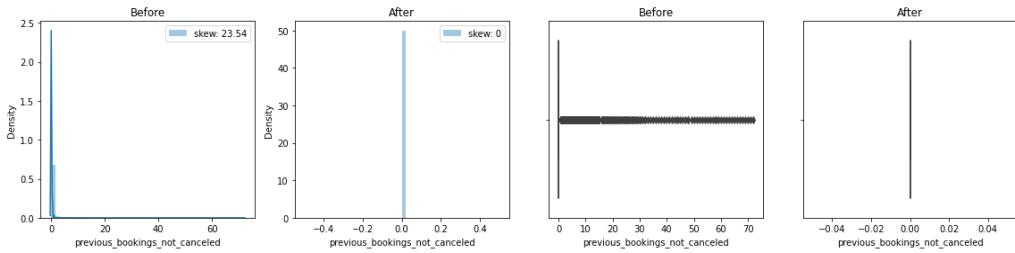
Hình 3.11: Thuộc tính stay_in_week_nights



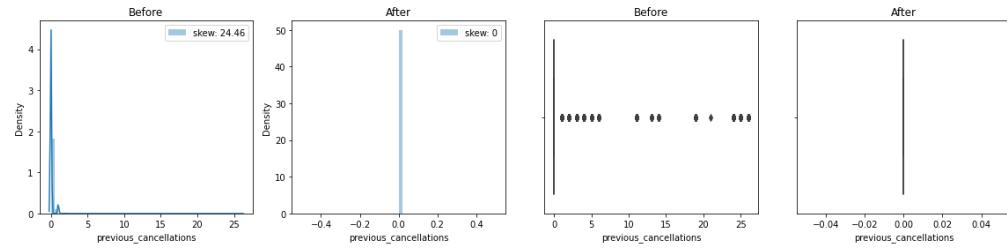
Hình 3.12: Thuộc tính adults



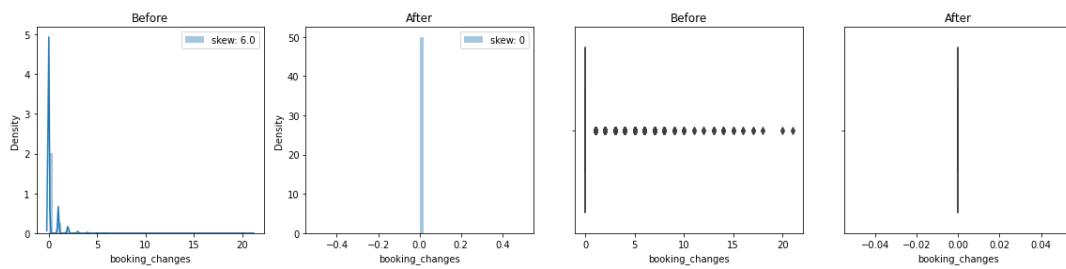
Hình 3.13: Thuộc tính babies



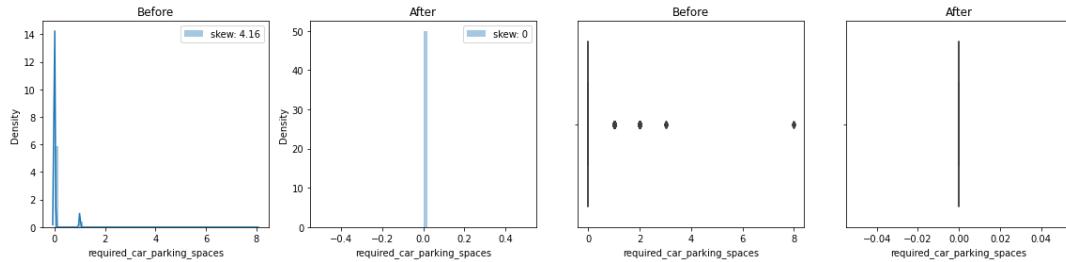
Hình 3.14: Thuộc tính previous_bookings_not_canceled



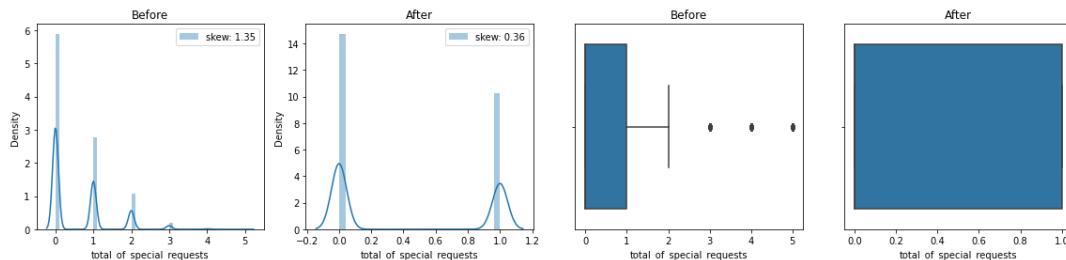
Hình 3.15: Thuộc tính previous_cancelations



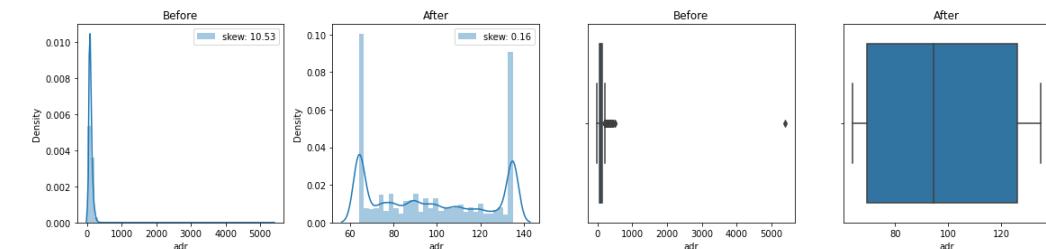
Hình 3.16: Thuộc tính booking_changes



Hình 3.17: Thuộc tính required_car_parking_spaces



Hình 3.18: Thuộc tính total_of_special_requests



Hình 3.19: Thuộc tính adr

- Nhận xét:

- Có một vài thuộc tính không trả về giá trị skew = 0 nên ta sẽ tiến hành chuẩn hóa dữ liệu các cột đó ở những bước sau.
- Các cột đó là: lead_time, stays_in_weekend_nights, stay_in_week_nights, total_of_special_requests, adr.

2.3. Nhận diện các dữ liệu bị thiếu.

- Tiến hành thống kê các dữ liệu Null trong tập dữ liệu
 - Đếm tổng số dữ liệu Null

```
[ ] df.isnull().sum().sum()
```

129425

Hình 3.20: Xuất ra tổng số dữ liệu Null là 129425.

- Kiểm tra các thuộc tính và số lượng phần tử Null trong các thuộc tính đó

Hình 3.21: Xuất ra thông tin cụ thể thông tin về phần tử Null.

- Thông tin tỷ lệ dữ liệu Null và tỷ lệ từng thuộc tính có dữ liệu Null trong tập dữ liệu.

```
[ ] (df.isnull().sum().sum() / (119390 * 36)) * 100
```

3.01125629356637

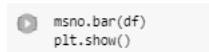
Hình 3.22: Tỷ lệ dữ liệu Null trên toàn tập dữ liệu

```
↳ [df.isnull().sum() / 119390] * 100
   hotel           0.00000
   is_canceled     0.00000
   lead_time        0.00000
   arrival_date_year 0.00000
   arrival_date_month 0.00000
   arrival_date_week_number 0.00000
   arrival_date_day_of_month 0.00000
   stays_in_weekend_nights 0.00000
   stays_in_week_nights 0.00000
   adults           0.00000
   children          0.003350
   babies            0.00000
   meal              0.00000
   country           0.48744
   market_segment     0.00000
   distribution_channel 0.00000
   is_repeated_guest 0.00000
   previous_cancellations 0.00000
   previous_bookings_not_canceled 0.00000
   reserved_room_type 0.00000
   assigned_room_type 0.00000
   booking_changes    0.00000
   deposit_type       0.00000
   agent              13.686238
   company            94.306893
   days_in_waiting_list 0.00000
   customer_type      0.00000
   adr                0.00000
   required_car_parking_spaces 0.00000
   total_of_special_requests 0.00000
   reservation_status 0.00000
   reservation_status_date 0.00000
   name               0.00000
   email              0.00000
   phone-number       0.00000
   credit_card         0.00000
dtype: float64
```

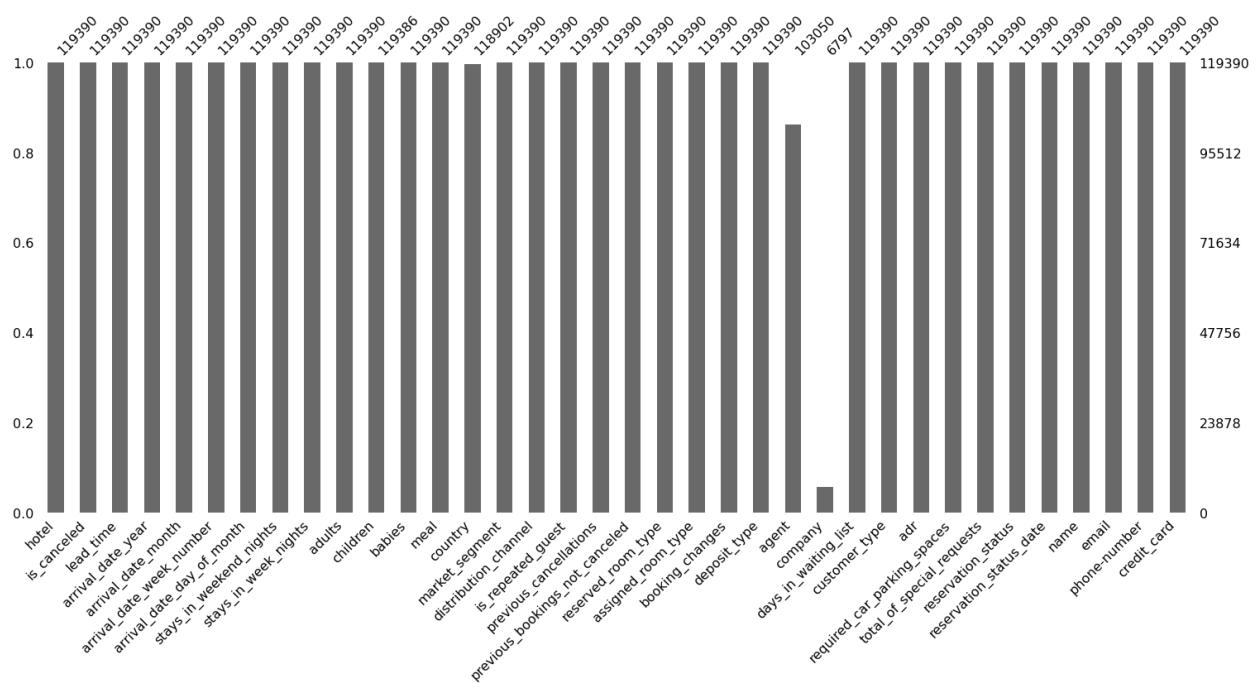
Hình 3.23: Tỷ lệ phần trăm từng thuộc tính có dữ liệu Null.

- Kết luận:

- Tập dữ liệu có tổng số dòng dữ liệu là 119 390 dòng và 36 cột. Trong đó, phần tử Null chiếm 3.01% số lượng dữ liệu trong hệ thống tương 129 425 phần tử có giá trị Null.
- Ngoài ra, các dữ liệu về thông tin cá nhân khách hàng và khách sạn để là dữ liệu ảo nhằm đảm bảo độ bảo mật thông tin.
- Biểu đồ đếm số giá trị khác Null trong từng thuộc tính:



Hình 3.24: Đoạn mã để vẽ biểu đồ thống kê thể hiện các dữ liệu khác Null.



Hình 3.25: Kết quả của biểu đồ cột.

2.4. Xử lý ý nghĩa của dữ liệu

- Xóa các khoảng trắng dư thừa trong các thuộc tính dạng chuỗi.

```
[701] features = ['meal', 'reserved_room_type', 'assigned_room_type', 'deposit_type', 'agent', 'company']
      for feature in features:
          df_win[feature] = df_win[feature].astype(str).str.strip()
```

Hình 3.26: Xóa bỏ các khoảng trống dư thừa.

- 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.

```
[645] filter = (df.children == 0) & (df.adults == 0) & (df.babies == 0)
      filter
```

```
0      False
1      False
2      False
3      False
4      False
       ...
119385  False
119386  False
119387  False
119388  False
119389  False
Length: 119390, dtype: bool
```

Hình 3.27 : Tạo dữ liệu có giá trị Boolean trả về kết quả True khi số lượng khách cho một lượt đặt phòng là 0, ngược lại trả về False.

```
df_win[filter]
```

hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	stays_in_week_nights	adults	...

0 rows x 36 columns

Hình 3.28: Xuất ra các dòng dữ liệu mà filter có giá trị là True và có 0 dòng dữ liệu có Filter là True.

2.5. Xử lý dữ liệu bị thiếu.

- Tìm và sắp xếp các dữ liệu Null.

```
[705] total = df_win.isnull().sum().sort_values(ascending=False)
```

Hình 3.29: Tìm trong tập dữ liệu giá trị Null, tính tổng chúng lại và sắp xếp theo thứ tự giảm dần.

```
[706] percent_1=df_win.isnull().sum()/df_win.isnull().count()*100
```

Hình 3.30: Tính phần trăm của tổng giá trị của Null trong một cột trên số dòng có giá trị Null của cột đó.

```
[650] percent_2 = (round(percent_1,1)).sort_values(ascending = False)
```

Hình 3.31: Làm tròn một chữ số và sắp xếp các giá trị phần trăm giảm dần.

```
missing_data = pd.concat([total,percent_2],axis=1,keys=['Total','%'])  
missing_data.head(5)
```

	Total	%
country	488	0.4
children	4	0.0
hotel	0	0.0
booking_changes	0	0.0
deposit_type	0	0.0

Hình 3.32: Xuất 5 thuộc tính có số lượng giá trị rỗng nhiều nhất.

- Nhận xét:

- Có hai thuộc tính chứa dữ liệu Null.
- Thuộc tính Children có 4 giá trị bị thiếu.
- Thuộc tính Country có 488 giá trị bị thiếu, chiếm hơn 10% dữ liệu.

- Giải pháp:
- Đối với thuộc tính Country, do có kiểu dữ liệu thuộc lớp Nominal nên ta dùng Mode để điền vào những ô còn trống.

```
[710] df_win['country'].describe()  
count      118902  
unique       177  
top        PRT  
freq      48590  
Name: country, dtype: object
```

Hình 3.32: Tìm kiếm thông tin có số lần xuất hiện cao nhất trong thuộc tính Country.

```
[711] df_win['country'].fillna(df_win['country'].mode().to_string(), inplace=True)
```

Hình 3.33: Thay thế vào vị trí trống giá trị của quốc giá xuất hiện nhiều nhất (PRT: Bồ Đào Nha).

- 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.

```
[711] df_win.dropna(subset=['children'], inplace=True)
```

Hình 3.34: Xóa các dòng dữ liệu Null trong thuộc tính Children.

2.6. Xử lý các giá trị NaN.

- Giải thích ý nghĩa của các dữ liệu trống:
 - Trên Kaggle có đề cập, tập dữ liệu này là dữ liệu thực của khách sạn nên tất cả các yếu tố liên quan đến việc nhận dạng khách sạn sẽ bị xóa và dữ liệu về khách hàng sẽ là các dữ liệu ảo nên các thuộc tính còn lại khi không có giá trị không phải là giá trị thiếu mà chỉ là không được sử dụng.
- Giải pháp:
 - Thay thế giá trị NaN trong cột Agent thành “No Agent”.

```
[712] df_win['agent'].replace(to_replace='nan', value='No Agent', inplace=True)
```

Hình 3.35: Thay thế giá trị NaN thành “No Agent”

- Thay thế giá trị Nan trong cột Company thành “No Company”

```
[713] df_win['company'].replace(to_replace='nan', value='No Company', inplace=True)
```

Hình 3.36: Thay thế giá trị NaN thành “No Company”

- Thay thế các giá trị Undenfined thành NaN.

-Giá trị Undefined không có ý nghĩa trong quá trình khai thác và xử lý dữ liệu về nhu cầu đặt phòng tại khách sạn nên ta xóa các giá trị Undefined đó, dữ liệu kiểu chuỗi rất quan trọng không thể thay thế như kiểu số bằng giá trị trung bình hay giá trị phổ biến nhất, Có hai cách để xử lý giá trị Undefined là xóa bỏ các dòng dữ liệu khỏi tập dữ liệu hoặc xem xét tỷ lệ các giá trị còn lại là bao nhiêu rồi thay thế Undefined theo đúng tỷ lệ đó.

- Biến đổi Undefined thành NaN và xóa các dòng dữ liệu.

```
df_win.replace('Undefined', np.nan, inplace=True)
```

Hình 3.37: Biến đổi dữ liệu từ Undefined thành NaN.

```
[719] df_win = df_win.dropna()
```

Hình 3.38: Xóa các dòng chứa NaN.

2.7. Kiểm tra thuộc tính của dữ liệu.

```
df_win.dtypes
```

Hình 3.39: Kiểm tra thuộc tính của dữ liệu.

↳ hotel	object
is_canceled	int64
lead_time	float64
arrival_date_year	int64
arrival_date_month	object
arrival_date_week_number	int64
arrival_date_day_of_month	int64
stays_in_weekend_nights	float64
stays_in_week_nights	float64
adults	float64
children	float64
babies	float64
meal	object
country	object
market_segment	object
distribution_channel	object
is_repeated_guest	int64
previous_cancellations	float64
previous_bookings_not_canceled	float64
reserved_room_type	object
assigned_room_type	object
booking_changes	float64
deposit_type	object
agent	object
company	object
days_in_waiting_list	float64
customer_type	object
adr	float64
required_car_parking_spaces	float64
total_of_special_requests	float64
reservation_status	object
reservation_status_date	object
name	object
email	object
phone-number	object
credit_card	object
dtype:	object

Hình 3.39: Kết quả kiểu dữ liệu của các thuộc tính.

- Nhận xét:

- Quan sát kiểu dữ liệu, ta nhận thấy kiểu dữ liệu có một số thuộc tính là float trong khi đó số lượng phải là một số tự nhiên nên thay đổi kiểu dữ liệu cho các thuộc tính này.

```
▶ df_win['children'] = df_win['children'].astype(int)
df_win['adults'] = df_win['adults'].astype(int)
df_win['babies'] = df_win['babies'].astype(int)
df_win['stays_in_weekend_nights'] = df_win['stays_in_weekend_nights'].astype(int)
df_win['stays_in_week_nights'] = df_win['stays_in_week_nights'].astype(int)
df_win['booking_changes'] = df_win['booking_changes'].astype(int)
df_win['days_in_waiting_list'] = df_win['days_in_waiting_list'].astype(int)
df_win['required_car_parking_spaces'] = df_win['required_car_parking_spaces'].astype(int)
df_win['total_of_special_requests'] = df_win['total_of_special_requests'].astype(int)
```

Hình 3.40: Thay đổi kiểu int cho các cột thuộc tính số.

- Tương tự, đối với thuộc tính reservation_status_date 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.

```
[716] df_win['reservation_status_date'] = pd.to_datetime(df_win['reservation_status_date'])
```

Hình 3.40: Thay đổi kiểu dữ liệu thành dạng ngày trong DataFrame.

2.8. Xử lý các dữ liệu bị trùng lặp.

```
df_win.duplicated().any()
```

Hình 3.41: Câu lệnh trả về True khi có dữ liệu bị trùng lặp.

```
False
```

Hình 3.42: Kết quả trả về là False tức là không có dữ liệu bị trùng.

3. Tích hợp dữ liệu

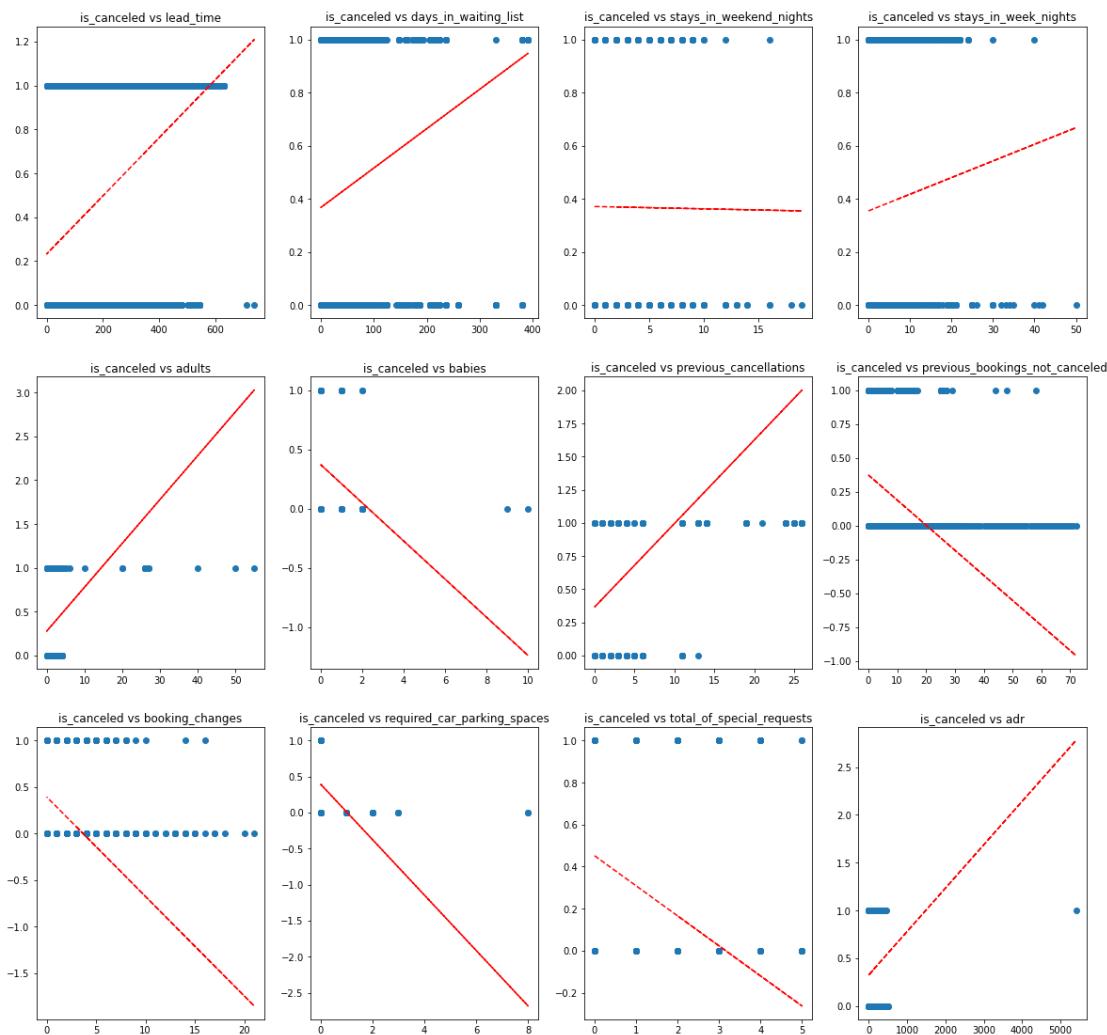
- Vẽ biểu đồ Scatter để hiện mức tương quan giữa biến phụ thuộc is_canceled với các thuộc tính độc lập khác

```
plt.figure(figsize=(20,40))

count = 1
for variable in cols:
    plt.subplot(6,4,count)
    x = df[variable]
    y = df["isCanceled"]
    z = np.polyfit(x, y, 1)
    p = np.poly1d(z)
    plt.scatter(x, y)
    plt.title("isCanceled vs " + variable)
    plt.plot(x,p(x),"r--")
    count = count + 1

plt.show()
```

Hình 3.43: Đoạn mã vẽ biểu đồ Scatter cho các thuộc tính định lượng.



Hình 3.44: Xuất ảnh biểu đồ Scatter theo 6 hàng và 4 cột.

- Nhận xét:

- Có năm thuộc tính có tính tương quan thuận với thuộc tính `is_canceled` là: `lead_time`, `days_in_waiting_list`, `stay_in_week_nights`, `adults`, `previous_cancellations`. Tức là khi giá trị của các biến độc lập tăng thì khả năng hủy đặt phòng càng cao.
- Ngược lại, các thuộc tính có tính tương quan nghịch như `babies`, `previous_bookings_not_canceled`, `booking_changes`, `required_car_parking_spaces` và `total_of_special_requests` có giá trị càng tăng thì khả năng nhận phòng càng cao.

- Thuộc tính stays_in_weekend_nights có mối tương quan khá thấp với biến phụ thuộc is_canceled.

4. Biến đổi dữ liệu và thu giảm dữ liệu

4.1. Thêm các thuộc tính.

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

```
[720] 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 dataset in [df_win]:
    dataset["arrival_date_month"] = dataset["arrival_date_month"].map(months)
```

Hình 3.45: Chuyển đổi tháng từ ký tự sang số.

```
[721] cols=["arrival_date_year","arrival_date_month","arrival_date_day_of_month"]
df_win['arrival_date_full'] = df_win[cols].apply(lambda x: '-'.join(x.values.astype(str)), axis="columns")
df_win['arrival_date_full'] = pd.to_datetime(df_win['arrival_date_full'])
```

Hình 3.46: Tạo thuộc tính arrival_date_full kiểu dữ liệu datetime thông qua ba thuộc tính: arrival_date_year, arrival_date_month và arrival_date_day_of_month.

- Tạo thuộc tính status_minus_arrival_date là hiệu số giữa arrival_date_full và reservation_status_date.

```
[726] df_win['status_minus_arrival_date'] = np.abs(df_win['arrival_date_full'] - df_win['reservation_status_date'])
def format_lenght(date):
    return date[0]
df_win['status_minus_arrival_date'] = df_win['status_minus_arrival_date'].map(format_lenght).astype(int)
```

Hình 3.47: Cú pháp tạo status_minus_arrival_date.

Trong đó:

- 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 is_family dựa trên các thuộc tính children, babies có số lượng nhiều hơn 0 trong tập dữ liệu.

```
[724] df_win['is_family'] = ((df_win['children'] != 0) | (df_win['babies'] != 0)) * 1
```

Hình 3.48: Tạo thuộc tính is_family.

- Tạo thuộc tính party_size là tổng số lượng người trong các đặc trưng adults, children, babies.

```
[723] df_win['party_size'] = df_win['adults'] + df_win['children'] + df_win['babies']
```

Hình 3.49: Tạo thuộc tính party_size.

- Tạo thuộc tính total_nights là tổng số đêm ở vào ngày cuối tuần (stay_in_weekend_nights) và ngày trong tuần (stay_in_week_nights).

```
[722] df_win['total_nights'] = df_win['stays_in_week_nights'] + df_win['stays_in_weekend_nights']
```

Hình 3.50: Tạo thuộc tính total_nights.

- Tạo thuộc tính room_difference là sự kết hợp giữa hai thuộc tính reserved_room_type và assigned_room_type chỉ ra liệu có sự khác nhau giữa số phòng đã đặt trước và số phòng nhận được của khách hàng hay không?

```
[722] df_win['total_nights'] = df_win['stays_in_week_nights'] + df_win['stays_in_weekend_nights']
```

Hình 3.51: Tạo thuộc tính room_difference.

4.2. Thu giảm dữ liệu

- Xóa các thuộc tính không cần thiết:
 - Xóa thuộc tính reservation_status, country, required_car_parking_spaces vì trong biểu đồ phân tích ở trên nghiên về một phía.
 - Xóa thuộc tính arrival_date_year, arrival_date_week_number, arrival_date_day_of_month vì đã dùng các thuộc tính này để tạo arrival_date_full.
 - Xóa thuộc tính arrival_date_full, reservation_status_date vì đã dùng các thuộc tính này để tạo status_minus_arrival_date.
 - Xóa thuộc tính reserved_room_type và assigned_room_type vì đã tạo thuộc tính room_difference.

- Xóa thuộc tính children, babies vì đã dùng các thuộc tính này để tạo thuộc tính party_size và is_family.
- Xóa thuộc tính agent và company vì chứa các dữ liệu không được định nghĩa cao.
- Xóa thuộc tính stay_in_weekend_nights và stay_in_week_nights vì dùng các thuộc tính này để tạo total_nights.

```
[728] df_win=df_win.drop(['reservation_status', 'days_in_waiting_list', 'required_car_parking_spaces', 'arrival_date_year', 'arrival_date_week_number', 'arrival_date_day_of_month', 'stays_in_weekend_nights', 'stays_in_week_nights', 'children', 'babies', 'country', 'reserved_room_type', 'assigned_room_type', 'agent', 'company', 'reservation_status_date', 'arrival_date_full'],axis=1)
```

Hình 3.52: Xóa các thuộc tính không cần thiết.

4.3. Chuẩn hóa dữ liệu.

- Thống kê các thuộc tính định lượng trong df_win.

	count	mean	std	min	25%	50%	75%	max
is_canceled	118216.0	0.371642	0.483245	0.0	0.0	0.0	1.0	1.0
lead_time	118216.0	86.254568	67.538980	11.0	18.0	69.0	161.0	184.0
arrival_date_month	118216.0	6.561295	3.076907	1.0	4.0	7.0	9.0	12.0
adults	118216.0	2.000000	0.000000	2.0	2.0	2.0	2.0	2.0
is_repeated_guest	118216.0	0.031958	0.175890	0.0	0.0	0.0	0.0	1.0
previous_cancellations	118216.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
previous_bookings_not_canceled	118216.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
booking_changes	118216.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
adr	118216.0	97.382738	27.024751	64.0	69.6	95.0	126.0	135.0
total_of_special_requests	118216.0	0.413785	0.492513	0.0	0.0	0.0	1.0	1.0
total_nights	118216.0	3.162254	1.572602	1.0	2.0	3.0	4.0	6.0
party_size	118216.0	2.104427	0.399613	2.0	2.0	2.0	2.0	12.0
is_family	118216.0	0.072291	0.258971	0.0	0.0	0.0	0.0	1.0
room_difference	118216.0	0.124535	0.330192	0.0	0.0	0.0	0.0	1.0
status_minus_arrival_date	118216.0	3.068087	2.133225	0.0	1.0	3.0	4.0	9.0

Hình 3.53: Thống kê các thông tin của thuộc tính kiểu số.

- Tiến hành chuẩn hóa dữ liệu bằng phương pháp Min-Max Normalization.

```
[730] def min_max_scaling(series):
    return (series - series.min()) / (series.max() - series.min())
```

Hình 3.54: Đoạn mã trực quan hóa bằng phương pháp Min-Max Normalization.

- Lựa chọn các thuộc tính cần chuẩn hóa và tiến hành chuẩn hóa dữ liệu.

```
[731] cols1 = ["lead_time", "adr", "status_minus_arrival_date", "total_nights"]

[732] for col in cols1:
    df_win[col] = min_max_scaling(df_win[col])
```

Hình 3.55: Lựa chọn thuộc tính và chuẩn hóa dữ liệu.

- Sau khi chuẩn hóa, độ lệch chuẩn của dữ liệu thay đổi, các thuộc tính (trừ arrival_date_month) có độ lệch chuẩn nhỏ hơn 1.

		count	mean	std	min	25%	50%	75%	max
	is_canceled	118216.0	0.371642	0.483245	0.0	0.000000	0.000000	1.000000	1.0
	lead_time	118216.0	0.434998	0.390399	0.0	0.040462	0.335260	0.867052	1.0
	arrival_date_month	118216.0	6.561295	3.076907	1.0	4.000000	7.000000	9.000000	12.0
	adults	118216.0	2.000000	0.000000	2.0	2.000000	2.000000	2.000000	2.0
	is_repeated_guest	118216.0	0.031958	0.175890	0.0	0.000000	0.000000	0.000000	1.0
	previous_cancellations	118216.0	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0
	previous_bookings_not_canceled	118216.0	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0
	booking_changes	118216.0	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0
	adr	118216.0	0.470179	0.380630	0.0	0.078873	0.436620	0.873239	1.0
	total_of_special_requests	118216.0	0.413785	0.492513	0.0	0.000000	0.000000	1.000000	1.0
	total_nights	118216.0	0.432451	0.314520	0.0	0.200000	0.400000	0.600000	1.0
	party_size	118216.0	2.104427	0.399613	2.0	2.000000	2.000000	2.000000	12.0
	is_family	118216.0	0.072291	0.258971	0.0	0.000000	0.000000	0.000000	1.0
	room_difference	118216.0	0.124535	0.330192	0.0	0.000000	0.000000	0.000000	1.0
	status_minus_arrival_date	118216.0	0.340899	0.237025	0.0	0.111111	0.333333	0.444444	1.0

Hình 3.56: Xuất thông tin tập dữ liệu sau khi chuẩn hóa.

5. Rời rạc hóa dữ liệu

- Thuộc tính lead_time.
 - Chuyển đổi thuộc tính lead_time về kiểu số nguyên và tạo ra các nhóm thời gian chia thời gian làm 4 phần nhỏ, sử dụng qcut để phân chia đổi ra ở mỗi nhóm (phương pháp làm mịn binning).

```
[734] df_win["lead_time"].describe()
   count    118216.000000
   mean      0.434998
   std       0.390399
   min       0.000000
   25%      0.040462
   50%      0.335260
   75%      0.867052
   max       1.000000
Name: lead_time, dtype: float64
```

Hình 3.57: Mô tả thuộc tính.

```
[735] test = pd.qcut(df_win["lead_time"], q=4)
```

Hình 3.58: Chia cột lead_time thành 4 giờ.

```
[736] test.value_counts()
```

(-0.001, 0.0405]	30053
(0.335, 0.867]	29518
(0.867, 1.0]	29356
(0.0405, 0.335]	29289

Name: lead_time, dtype: int64

Hình 3.59: Đếm giá trị thuộc tính mỗi giờ và hiển thị mỗi giờ có bao nhiêu giá trị.

```
[737] for dataset in [df_win]:
    dataset.loc[dataset['lead_time']<=0.0405, 'lead_time']=1
    dataset.loc[(dataset['lead_time']>0.0405) & (dataset['lead_time']<=0.335), 'lead_time']=2
    dataset.loc[(dataset['lead_time']>0.335) & (dataset['lead_time']<=1.0], 'lead_time']=3
    dataset.loc[dataset['lead_time']>1.0, 'lead_time']=4
    dataset['lead_time']=dataset['lead_time'].astype(int)
```

Hình 3.60: Chia giờ thành 4 khoảng.

- 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).

```
[738] df_win["adr"].describe()
   count    118216.000000
   mean      0.470179
   std       0.380630
   min       0.000000
   25%      0.078873
   50%      0.436620
   75%      0.873239
   max       1.000000
Name: adr, dtype: float64
```

Hình 3.61: Mô tả thuộc tính.

```
✓ [739] test = pd.qcut(df_win["adr"], q=4)
```

Hình 3.62: Chia cột lead_time thành 4 giờ.

```
[740] test.value_counts()
```

(0.0789, 0.437]	31043
(-0.001, 0.0789]	29556
(0.873, 1.0]	28988
(0.437, 0.873]	28629
Name: adr, dtype: int64	

Hình 3.63: Đếm giá trị thuộc tính mỗi giờ và hiển thị mỗi giờ có bao nhiêu giá trị.

```
[741] for dataset in [df_win]:
    dataset.loc[dataset['adr']<=0.0789, 'adr']=1
    dataset.loc[(dataset['adr']>0.0789) & (dataset['adr']<=0.437), 'adr']=2
    dataset.loc[(dataset['adr']>0.437) & (dataset['adr']<=0.873), 'adr']=3
    dataset.loc[dataset['adr']>0.873, 'adr']=4
    dataset['adr']=dataset['adr'].astype(int)
```

Hình 3.64: Chia giờ thành 4 khoảng.

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

```
[742] df_win["total_nights"].describe()
```

count	118216.000000
mean	0.432451
std	0.314520
min	0.000000
25%	0.200000
50%	0.400000
75%	0.600000
max	1.000000
Name: total_nights, dtype: float64	

Hình 3.65: Mô tả thuộc tính.

```
[743] test = pd.qcut(df_win["total_nights"], q=4)
```

Hình 3.66: Chia cột lead_time thành 4 giờ.

```
[744] test.value_counts()
```

(-0.001, 0.2]	46869
(0.2, 0.4]	28831
(0.6, 1.0]	24386
(0.4, 0.6]	18130
Name: total_nights, dtype: int64	

Hình 3.67: Đếm giá trị thuộc tính mỗi giờ và hiển thị mỗi giờ có bao nhiêu giá trị.

```
[1]: for dataset in [df_train]:
    dataset.loc[dataset['total_nights']<=0.2, 'total_nights']=1
    dataset.loc[(dataset['total_nights']>0.2) & (dataset['total_nights']<=0.4), 'total_nights']=2
    dataset.loc[(dataset['total_nights']>0.4) & (dataset['total_nights']<=0.6), 'total_nights']=3
    dataset.loc[dataset['total_nights']>0.6, 'total_nights']=4
    dataset['total_nights']=dataset['total_nights'].astype(int)
```

Hình 3.68: Chia gio thành 4 khoảng.

- Thuộc tính status_minus_arrival_date.

- Chuyển đổi thuộc tính status_minus_arrival_date về kiểu số nguyên và tạo ra các nhóm thời gian chia thời gian làm 3 phần nhỏ, sử dụng qcut để phân chia đổi ra ở mỗi nhóm (phương pháp làm mịn binning).

```
[746]: df_train["status_minus_arrival_date"].describe()
```

	count	mean	std	min	25%	50%	75%	max
	118216.000000	0.340899	0.237025	0.000000	0.111111	0.333333	0.444444	1.000000
Name:	status_minus_arrival_date	dtype:	float64					

Hình 3.69: Mô tả thuộc tính.

```
[1]: test = pd.qcut(df_train["status_minus_arrival_date"], q=3)
```

Hình 3.70: Chia cột lead_time thành 3 giờ.

```
[1]: test.value_counts()
```

Bin	Count
(-0.001, 0.222]	57905
(0.222, 0.444]	34644
(0.444, 1.0]	25667
Name:	status_minus_arrival_date
dtype:	int64

Hình 3.71: Đếm giá trị thuộc tính mỗi giờ và hiển thị mỗi giờ có bao nhiêu giá trị.

```
[1]: for dataset in [df_train]:
    dataset.loc[dataset['status_minus_arrival_date']<=0.222, 'status_minus_arrival_date']=1
    dataset.loc[(dataset['status_minus_arrival_date']>0.222) & (dataset['status_minus_arrival_date']<=0.444), 'status_minus_arrival_date']=2
    dataset.loc[dataset['status_minus_arrival_date']>0.444, 'status_minus_arrival_date']=3
    dataset['status_minus_arrival_date']=dataset['status_minus_arrival_date'].astype(int)
```

Hình 3.72: Chia gio thành 3 khoảng.

6. Sampling

- Giải thích:
 - Do dữ liệu về nhân phòng và hủy phòng chênh lệch quá cao có thể gây ra thiên vị trong quá trình xây dựng mô hình nên ta sẽ lọc ngẫu nhiên các dữ liệu nhận phòng và lấy ra ngẫu nhiên số lượng dữ liệu bằng với số lượng dữ liệu trong hủy phòng.
- Thực hiện:



```
# Shuffle the Dataset.
shuffled_df = df_win.sample(frac=1,random_state=4)

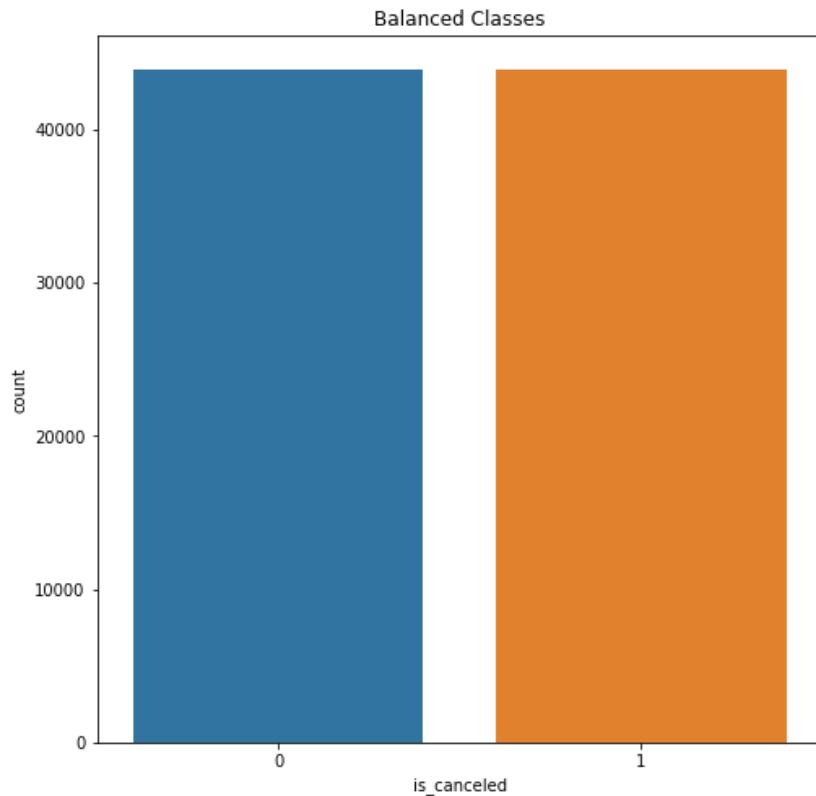
# Put all the fraud class in a separate dataset.
Canceled_df = shuffled_df.loc[shuffled_df['is_canceled'] == 1]

#Randomly select 492 observations from the non-fraud (majority class)
NonCanceled_df = shuffled_df.loc[shuffled_df['is_canceled'] == 0].sample(n=df_win["is_canceled"].value_counts()[1],random_state=42)

# Concatenate both dataframes again
sampling_win_df = pd.concat([Canceled_df, NonCanceled_df])

#plot the dataset after the undersampling
plt.figure(figsize=(8, 8))
sns.countplot('is_canceled', data=sampling_win_df)
plt.title('Balanced Classes')
plt.show()
```

Hình 3.73: Lấy các dòng dữ liệu hủy phòng và nhận phòng lần lượt gán vào hai dataframe `Canceled_df` và `NonCancelled_df` với số lượng dữ liệu hai dataframe bằng nhau và gộp dữ liệu vào trong cùng dataframe tên `sampling_win_df` và hiển thị đồ thị `CountPlot` thể hiện số lượng dữ liệu trong mỗi dataframe con.



Hình 3.74: Kết quả khi chạy biểu đồ CountPlot.

7. Biểu diễn dữ liệu

- Xem lại thông tin tập dữ liệu.

```
▶ df_win.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 118216 entries, 0 to 119389
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   hotel            118216 non-null   object 
 1   is_canceled      118216 non-null   int64  
 2   lead_time         118216 non-null   int64  
 3   arrival_date_month 118216 non-null   int64  
 4   adults            118216 non-null   int64  
 5   meal              118216 non-null   object 
 6   market_segment     118216 non-null   object 
 7   distribution_channel 118216 non-null   object 
 8   is_repeated_guest 118216 non-null   int64  
 9   previous_cancellations 118216 non-null   float64
 10  previous_bookings_notCanceled 118216 non-null   float64
 11  booking_changes    118216 non-null   int64  
 12  deposit_type       118216 non-null   object 
 13  customer_type      118216 non-null   object 
 14  adr                118216 non-null   int64  
 15  total_of_special_requests 118216 non-null   int64  
 16  total_nights        118216 non-null   int64  
 17  party_size          118216 non-null   int64  
 18  is_family           118216 non-null   int64  
 19  room_difference     118216 non-null   int64  
 20  status_minus_arrival_date 118216 non-null   int64  
dtypes: float64(2), int64(13), object(6)
memory usage: 19.8+ MB
```

Hình 3.75: Xem thông tin tổng quát của tập dữ liệu.

- Xem kiểu dữ liệu của thuộc tính.

```
[931]: df_wd.head()
```

	hotel	is_canceled	lead_time	arrival_date_month	adults	meal	market_segment	distribution_channel	is_repeated_guest	previous_cancellations	...	booking_changes	deposit_type	customer_type	adr	tot
0	Resort Hotel	0	4		7	2	BB	Direct	Direct	0	0.0	...	0	No Deposit	Transient	4
1	Resort Hotel	0	4		7	2	BB	Direct	Direct	0	0.0	...	0	No Deposit	Transient	4
2	Resort Hotel	0	4		7	2	BB	Direct	Direct	0	0.0	...	0	No Deposit	Transient	4
3	Resort Hotel	0	4		7	2	BB	Corporate	Corporate	0	0.0	...	0	No Deposit	Transient	4
4	Resort Hotel	0	4		7	2	BB	Online TA	TA/TO	0	0.0	...	0	No Deposit	Transient	4

5 rows × 21 columns

Hình 3.76: Xem thông tin hiển thị 5 dòng thuộc tính đầu tiên.

8. Thu giảm số chiều dữ liệu.

- Hiển thị đồ thị heatmap để thể hiện sự mối tương quan của các thuộc tính.



Hình 3.77: Thể hiện mối độ tương quan trong đồ thị

- Tạo hàm xóa các thuộc tính có mối tương quan cao.

```
def correlation(dataset, threshold):
    col_corr = set() #Tập hợp tất cả các cột tương quan
    corr_matrix = dataset.corr()
    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
```

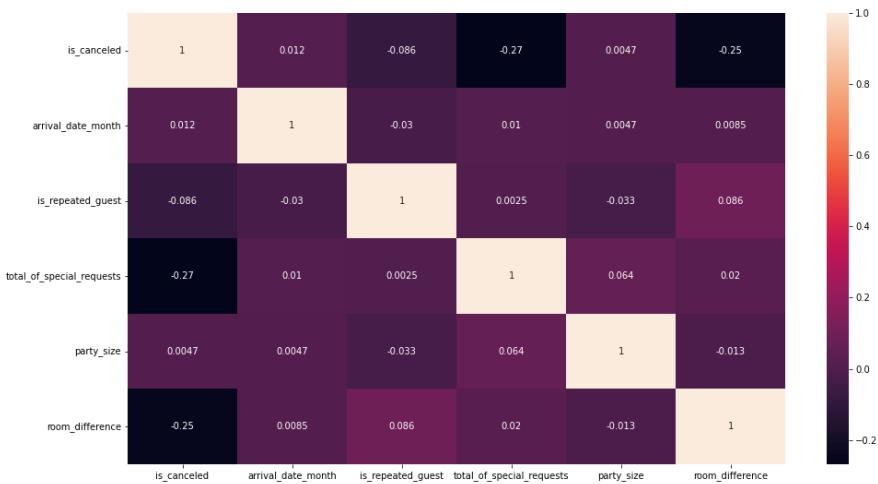
Hình 3.78: Tạo hàm loại bỏ các thuộc tính có độ tương quan cao (bao gồm thuận và nghịch).

- Loại bỏ các thuộc tính có mối tương quan cao.

```
df1 = df.copy()
c_features = ['lead_time', 'adults', 'previous_cancellations', 'previous_bookings_not_canceled',
              'booking_changes', 'adr', 'total_nights', 'status_minus_arrival_date']
df1 = df1.drop(c_features, axis = 1)
corr_features = correlation(df1, corr)
print("-----With corr equal: {0}-----".format(corr))
print("Length of corr_feature equal: {0}".format(len(set(corr_features))))
df1 = df1.drop(corr_features, axis = 1)
plt.figure(figsize=(16,9))
sns.heatmap(df1.corr(method='pearson'), annot=True)
```

Hình 3.79: Loại bỏ các khoảng trống và loại bỏ thuộc tính có độ tương quan cao hơn 0.7.

- Kiểm tra lại đồ thị heatmap sau loại bỏ các thuộc tính có mối tương quan cao.

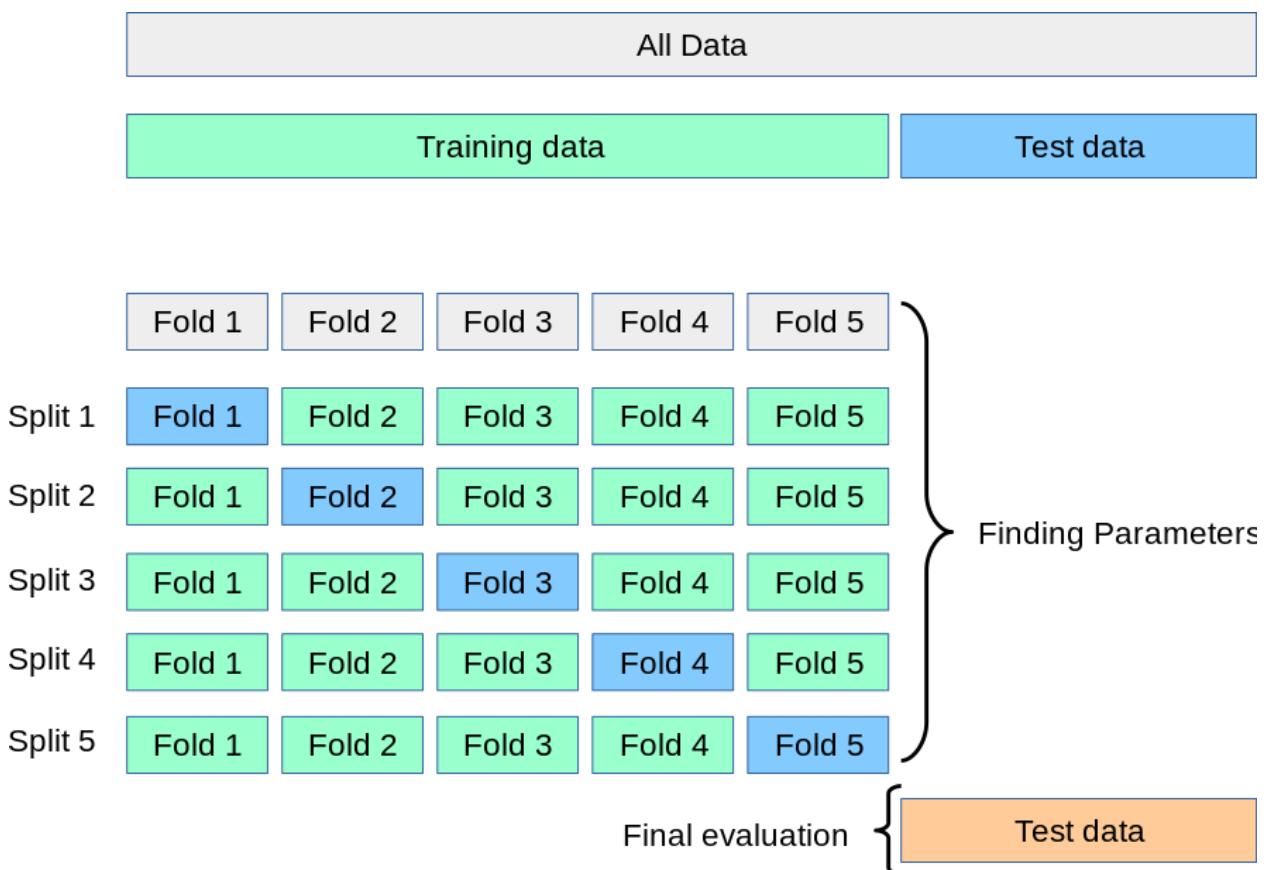


Hình 3.80: Kết quả thu được.

CHƯƠNG 4: ỨNG DỤNG GIẢI THUẬT PHÂN LỚP VÀO TẬP DỮ LIỆU

1. K-Fold Cross Validation

- Cross Validation là phương pháp thống kê được sử dụng để ước lượng hiệu quả các mô hình học máy. Được sử dụng để so sánh và chọn ra mô hình tốt nhất cho một bài toán.
- K-Fold Cross Validation là kỹ thuật lấy mẫu để đánh giá mô hình học máy trong k lần.
- Ví dụ minh họa:



Hình 4.1: Mô tả các hoạt động của k-Fold Cross Validation.

- Ứng dụng trong đồ án:

```
folds = ShuffleSplit(n_splits=7, random_state=42, test_size=0.3)
feature_onehot = np.array(feature_onehot)
label = np.array(label)
i = 1
for train_index, test_index in folds.split(label):
    print("-----Time: {}".format(i))
    x_train, x_test = feature_onehot[train_index], feature_onehot[test_index]
    y_train, y_test = label[train_index], label[test_index]
    get_score(x_train, x_test, y_train, y_test, i)
    i = i + 1
```

Hình 4.2: Code trong đồ án.

- Giải thích ý nghĩa:

- Hàm ShufleSplit dùng để tách dữ liệu 7 lần trong mỗi lần tách sẽ có 70% là dữ liệu để train (huấn luyện) và 30% dữ liệu để test (kiểm định).
- Tách dữ liệu thành hai phần: dữ liệu bình thường (feature_onehot) và dữ liệu chứa thuộc tính quyết định (label).
- Hàm for dùng để truy cập từng dòng trong mỗi lần tách đó và gán các index (chỉ mục) tương ứng vào trong tập huấn luyện và kiểm định rồi tiến hành xây dựng các mô hình.

2. Baseline Model

2.1. Khái niệm Baseline Model

Baseline model là một model đơn giản được sử dụng để so sánh các model hoạt động tốt như thế nào. Baseline giúp những nhà lập trình định lượng được hiệu năng tối thiểu, dự kiến về một vấn đề cụ thể.

2.2. Ưu điểm

Một baseline chỉ mất 10% thời gian để phát triển, nhưng sẽ giúp ta đạt được 90% chặng đường để đạt được kết quả hợp lý do baseline đóng vai trò là tiêu chuẩn cho các model được đào tạo.

Dễ triển khai, đào tạo nhanh hơn vì một số tham số có thể nhanh chóng phù hợp với dữ liệu của bạn và nó thường đủ đơn giản để cho phép dễ dàng phát hiện sự cố.

Cải thiện hiểu biết về dữ liệu, baseline có thể cho biết liệu tập dữ liệu có đủ hoặc không đủ cho tác vụ học máy muốn tiến hành hay không.

2.3. Giải thuật Baseline

- Kiểm định thời gian chạy của thuật toán

```
start= time.time()
bm = DummyClassifier(strategy='most_frequent', random_state=0)
bm.fit(x_train, y_train)
end = time.time()
```

Hình 4.3: Đoạn code lưu lại thời gian chạy.

- Chạy thuật toán bằng những dòng lệnh sau

```
print("-----Baseline Model-----")
start= time.time()
bm = DummyClassifier(strategy='most_frequent', random_state=0)
bm.fit(x_train, y_train)
end = time.time()
time_bm.append(end - start)
training_score_bm.append(bm.score(x_train, y_train))
bm_pred = bm.predict(x_test)
bm_score = metrics.accuracy_score(y_test, bm_pred)
test_score_bm.append(bm_score)
print("Baseline Model")
print("Accuracy:",bm_score)
print("Report:",metrics.classification_report(y_test,bm_pred))
bm_cm = metrics.confusion_matrix(y_test,bm_pred)
```

Hình 4.4: Đoạn code của Baseline Model.

- Tính toán ma trận nhầm lẫn:

```
-----Baseline Model-----
Baseline Model
Accuracy: 0.6261666431693219
Report:
precision    recall   f1-score   support
          0       0.63      1.00      0.77     22207
          1       0.00      0.00      0.00     13258

accuracy                           0.63      35465
macro avg       0.31      0.50      0.39     35465
weighted avg    0.39      0.63      0.48     35465
```

Hình 4.5: Kết quả ma trận nhầm lẫn trong lần chạy đầu tiên.

- Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

```
print("-----1. Baseline Model-----")
print("Time: {}".format(time_bm))
print("Training Accuracy: {}".format(training_score_bm))
print("Test Accuracy: {}".format(test_score_bm))

-----
1. Baseline Model-----
Time: [0.0030150413513183594, 0.004001140594482422, 0.0030012130737304688, 0.0031859874725341797, 0.0030007362365722656, 0.0, 0.0]
Training Accuracy: [0.6292975311476598, 0.6282703532283598, 0.6293579533782069, 0.6278232287223116, 0.6269048108179962, 0.6292250244710034, 0.6289833355488151]
Test Accuracy: [0.6261666431693219, 0.6285633723389257, 0.6260256591005217, 0.6296066544480474, 0.6317496122938108, 0.6263358240518822, 0.626899760327083]
```

Hình 4.5: Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

3. Decision Tree Model

3.1. Khái niệm

- Mô hình cây quyết định là một loại máy học có giám sát được sử dụng để phân loại hoặc đưa ra dự đoán dựa trên cách trả lời một bộ câu hỏi trước đó được sử dụng cho cả nhiệm vụ phân loại và hồi quy và hoạt động đối với dữ liệu và biến số hoặc phân loại.
- Có hai thuật toán nổi bật có thể tạo ra mô hình cây quyết định :
 - Decision Tree Cart : là thuật ngữ chỉ một mô hình cây trong đó biến mục tiêu có thể nhận một bộ giá trị rời rạc được gọi là cây phân loại. và cây quyết định trong đó biến mục tiêu là các giá trị liên tục (thường là số thực) được gọi là cây hồi quy.
 - Decision Tree ID3: là viết tắt của Iterative Dichotomiser 3 và được đặt tên như vậy vì thuật toán lặp đi lặp lại (lặp đi lặp lại) phân đôi (chia) các tính năng thành hai hoặc nhiều nhóm ở mỗi bước.

3.2. Ưu điểm

- Dễ hiểu ngay cả đối với những người không có kiến thức nền tảng về phân tích.
- Yêu cầu ít làm sạch dữ liệu hơn các kỹ thuật mô hình hóa dữ liệu khác.
- Thể hiện một cách trực quan các mối quan hệ nguyên nhân và kết quả, cung cấp một cái nhìn đơn giản về các quá trình phức tạp từ đó làm rõ rู้ ro, mục tiêu và lợi ích.

- Do cấu trúc lưu đồ đơn giản của cấu trúc cây, đây là một trong những phương pháp nhanh nhất để xác định các biến quan trọng và mối quan hệ giữa hai hoặc nhiều biến.

3.3. Decision Tree ID3 Model

3.3.1. Khái niệm Baseline Model

Baseline model là một model đơn giản được sử dụng để so sánh các model hoạt động tốt như thế nào. Baseline giúp những nhà lập trình định lượng được hiệu năng tối thiểu, dự kiến về một vấn đề cụ thể.

3.3.2. Ưu điểm

Một baseline chỉ mất 10% thời gian để phát triển, nhưng sẽ giúp ta đạt được 90% chặng đường để đạt được kết quả hợp lý do baseline đóng vai trò là tiêu chuẩn cho các model được đào tạo.

Dễ triển khai, đào tạo nhanh hơn vì một số tham số có thể nhanh chóng phù hợp với dữ liệu của bạn và nó thường đủ đơn giản để cho phép dễ dàng phát hiện sự cố.

Cải thiện hiểu biết về dữ liệu, baseline có thể cho biết liệu tập dữ liệu có đủ hoặc không đủ cho tác vụ học máy muốn tiến hành hay không.

3.3.3. Giải thuật Decision Tree ID3

- Kiểm định thời gian chạy của thuật toán

```
start= time.time()
id3 = tree.DecisionTreeClassifier(criterion="entropy", random_state=0)
id3.fit(x_train,y_train)
end = time.time()
```

Hình 4.6: Đoạn code lưu lại thời gian chạy.

- Chạy thuật toán bằng những dòng lệnh sau:

```
print("-----Decision Tree ID3-----")
start= time.time()
id3 = tree.DecisionTreeClassifier(criterion="entropy",random_state=0)
id3.fit(x_train,y_train)
end = time.time()
time_id3.append(end - start)
training_score_id3.append(id3.score(x_train, y_train))
id3_pred = id3.predict(x_test)
id3_score = metrics.accuracy_score(y_test, id3_pred)
test_score_id3.append(id3_score)
print("Decision Tree ID3")
print("Accuracy:",id3_score)
print("Report:",metrics.classification_report(y_test,id3_pred))
id3_cm = metrics.confusion_matrix(y_test,id3_pred)
```

Hình 4.7: Đoạn code của Decision Tree ID3 Model.

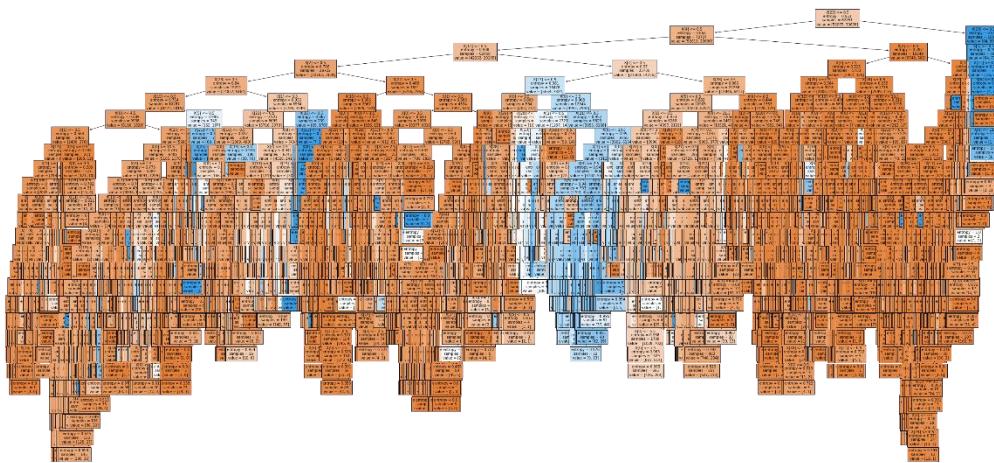
- Tính toán ma trận nhầm lẫn:

```
-----Decision Tree ID3-----
Decision Tree ID3
Accuracy: 0.7924432539123079
Report:
precision    recall   f1-score   support
          0       0.80      0.90      0.84     22207
          1       0.78      0.61      0.69     13258

accuracy                   0.79      35465
macro avg        0.79      0.76      0.77      35465
weighted avg     0.79      0.79      0.79      35465
```

Hình 4.8: Kết quả ma trận nhầm lẫn trong lần chạy đầu tiên.

- Mô hình cây quyết định:



Hình 4.9: Mô hình cây quyết định ID3 trong lần chạy đầu tiên.

- Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

```

print("-----2. Decision Tree ID3-----")
print("Time: {}".format(time_id3))
print("Training Accuracy: {}".format(training_score_id3))
print("Test Accuracy: {}".format(test_score_id3))

-----2. Decision Tree ID3-----
Time: [0.2656078338623847, 0.4061555862426758, 0.22623705863952637, 0.2555983066558838, 0.2239995002746582, 0.2263023853302002, 0.21963715553283691]
Training Accuracy: [0.7981414121883723, 0.7992652656765478, 0.7991081678771254, 0.7985522833560924, 0.7989269011854842, 0.7992773501226571, 0.8001353457964254]
Test Accuracy: [0.7924432539123079, 0.7885238967996616, 0.7897645566051036, 0.7925560411673481, 0.790018327928944, 0.7892288171436628, 0.788157338220781]

```

Hình 4.10: Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

3.4. Decision Tree Cart Model

- Kiểm định thời gian chạy của thuật toán

```

start= time.time()
cart = tree.DecisionTreeClassifier(criterion="gini", random_state=0)
cart.fit(x_train,y_train)
end = time.time()

```

Hình 4.11: Đoạn code lưu lại thời gian chạy.

- Chạy thuật toán bằng những dòng lệnh sau:

```
print("-----Decision Tree Cart-----")
start= time.time()
cart = tree.DecisionTreeClassifier(criterion="gini",random_state=0)
cart.fit(x_train,y_train)
end = time.time()
time_cart.append(end - start)
training_score_cart.append(cart.score(x_train, y_train))
cart_pred = cart.predict(x_test)
cart_score = metrics.accuracy_score(y_test, cart_pred)
test_score_cart.append(cart_score)
print("Decision Tree Cart")
print("Accuracy:",cart_score)
print("Report:",metrics.classification_report(y_test,cart_pred))
cart_cm = metrics.confusion_matrix(y_test,cart_pred)
save_tree_cart(cart, i)
```

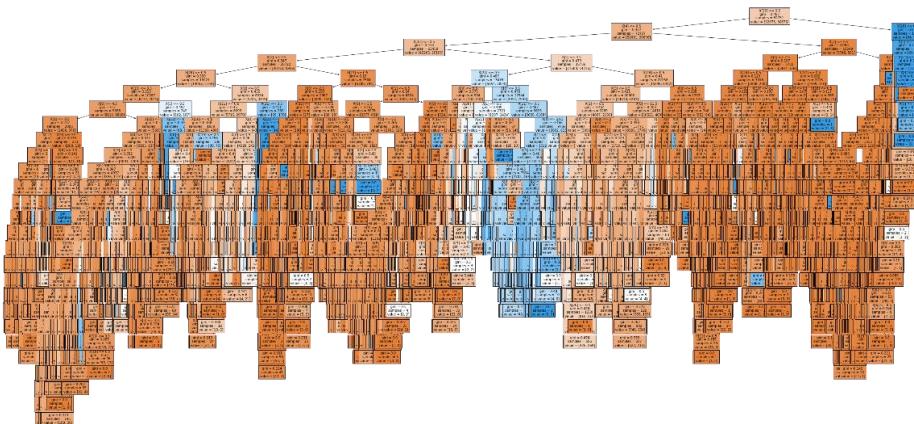
Hình 4.12: Đoạn code của Decision Tree Cart Model.

- Tính toán ma trận nhầm lẫn:

```
-----Decision Tree Cart-----
Decision Tree Cart
Accuracy: 0.7920766953334273
Report:
precision      recall   f1-score   support
          0       0.80      0.90      0.84     22207
          1       0.78      0.61      0.69     13258
accuracy
macro avg       0.79      0.76      0.77     35465
weighted avg     0.79      0.79      0.79     35465
```

Hình 4.13: Kết quả ma trận nhầm lẫn trong lần chạy đầu tiên.

- Mô hình cây quyết định:



Hình 4.14: Mô hình cây quyết định Cart trong lần chạy đầu tiên.

- Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

```
print("-----3. Decision Tree Cart-----")
print("Time: {}".format(time_cart))
print("Training Accuracy: {}".format(training_score_cart))
print("Test Accuracy: {}".format(test_score_cart))

-----
3. Decision Tree Cart-----
Time: [0.22311806678771973, 0.27906298637390137, 0.21821141242980957, 0.23026442527770996, 0.2636098861694336, 0.23679685592651367, 0.26123523712158203]
Training Accuracy: [0.7981414121883723, 0.7992652656765478, 0.7991081678771254, 0.7985522833560924, 0.798269011854842, 0.7992773501226571, 0.8001353457964254]
Test Accuracy: [0.7920766953334273, 0.788749471309742, 0.7898491470463838, 0.7922458762159876, 0.7901311151839842, 0.7890878330748625, 0.7881291414070211]
```

Hình 4.15: Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

4. Naïve Bayes Model

4.1. Khái niệm Naïve Bayes Model

- Naïve Bayes là một thuật toán phân lớp được mô hình hoá dựa trên định lý Bayes trong xác suất thống kê, được sử dụng trong nhiều nhiệm vụ phân loại.

4.2. Ưu điểm

- Đơn giản và dễ thực hiện.
- Không yêu cầu nhiều dữ liệu đào tạo nên thời gian huấn luyện sẽ ngắn hơn và có thể xử lý cả dữ liệu liên tục và dữ liệu rời rạc.

- Dễ dàng và nhanh chóng dự đoán lớp của tập dữ liệu thử nghiệm trong thời gian thực. Nó cũng hoạt động tốt trong dự đoán nhiều lớp.
- Khi giả định độc lập, Naive Bayes hoạt động tốt hơn so với các mô hình khác như hồi quy logistic.

4.3. Giải thuật Naïve Bayes

- Kiểm định thời gian chạy của thuật toán

```
start= time.time()
nb = GaussianNB()
nb.fit(x_train,y_train)
end = time.time()
```

Hình 4.16: Đoạn code lưu lại thời gian chạy.

- Chạy thuật toán bằng những dòng lệnh sau

```
print("----Naive Bayes----")
start= time.time()
nb = GaussianNB()
nb.fit(x_train,y_train)
end = time.time()
time_nb.append(end - start)
training_score_nb.append(nb.score(x_train, y_train))
nb_pred = nb.predict(x_test)
nb_score = metrics.accuracy_score(y_test, nb_pred)
test_score_nb.append(nb_score)
print("Naive Bayes")
print("Accuracy:",nb_score)
print("Report:",metrics.classification_report(y_test,nb_pred))
nb_cm = metrics.confusion_matrix(y_test,nb_pred)
```

Hình 4.17: Đoạn code của Naïve Bayes Model.

- Tính toán ma trận nhầm lẫn:

```
----Naive Bayes-----
Naive Bayes
Accuracy: 0.6795150148033272
Report:
precision      recall   f1-score   support
          0       0.81      0.64      0.71    22207
          1       0.55      0.75      0.64    13258

accuracy           0.68      0.68      0.68    35465
macro avg         0.68      0.69      0.68    35465
weighted avg      0.71      0.68      0.68    35465
```

Hình 4.18: Kết quả ma trận nhầm lẫn trong lần chạy đầu tiên.

- Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

```
print("-----4. Naive Bayes-----")
print("Time: {}".format(time_nb))
print("Training Accuracy: {}".format(training_score_nb))
print("Test Accuracy: {}".format(test_score_nb))

-----
-----4. Naive Bayes-----
Time: [0.03822779655456543, 0.03815484046936035, 0.03740072250366211, 0.04020357131958008, 0.029996633529663086, 0.046060800552368164, 0.047011375427246094]
Training Accuracy: [0.6732849149859216, 0.6650312382931929, 0.6722335681744027, 0.6654904472453506, 0.6722214837282933, 0.6712305591473214, 0.6759193242377736]
Test Accuracy: [0.6795150148033272, 0.6603975750740166, 0.6711687579303539, 0.6658113633159453, 0.6700408853799521, 0.6694487522909911, 0.6745241787677799]
```

Hình 4.19: Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

5. Random Forest Model

5.1. Khái niệm Random Forest Model

- Random forest là một mô hình học máy được giám sát được sử dụng rộng rãi trong các bài toán Phân loại và Hồi quy. Nó xây dựng cây quyết định trên các mẫu khác nhau và lấy một trường hợp có đa số phiếu bầu của các cây quyết định này làm kết quả dự đoán và nó có thể được sử dụng cho cả nhiệm vụ phân loại và hồi quy.

5.2. Ưu điểm

- Một trong những lợi thế lớn nhất của random forest là tính linh hoạt của nó. Nó có thể được sử dụng cho cả nhiệm vụ hồi quy và phân loại, đồng thời cũng dễ dàng xem được tầm quan trọng tương đối mà nó gán cho các tính năng đầu vào.
- Là một mô hình tiện dụng vì các siêu tham số mặc định mà nó sử dụng thường tạo ra kết quả dự đoán tốt.
- Một trong vấn đề lớn nhất trong học máy là tình trạng overfitting (hiện tượng mô hình dự đoán quá khớp với tập training set, dẫn đến dự đoán không hiệu quả đối với tập testing set), tuy nhiên nhờ vào bộ phân loại của Random Forest thì tình trạng này hiếm khi xảy ra khi mà có đủ tree trong forest.

5.3. Giải thuật Random Forest

- Kiểm định thời gian chạy của thuật toán

```
start= time.time()
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
end = time.time()
```

Hình 4.20: Đoạn code lưu lại thời gian chạy.

- Chạy thuật toán bằng những dòng lệnh sau

```
print("-----Random Forest-----")
start= time.time()
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
end = time.time()
time_rf.append(end - start)
training_score_rf.append(rf.score(x_train, y_train))
rf_pred = rf.predict(x_test)
rf_score = metrics.accuracy_score(y_test, rf_pred)
test_score_rf.append(rf_score)
print("Random Forest")
print("Accuracy:",rf_score)
print("Report:",metrics.classification_report(y_test,rf_pred))
rf_cm = metrics.confusion_matrix(y_test,rf_pred)
save_randomForest(rf, i, x_train)
```

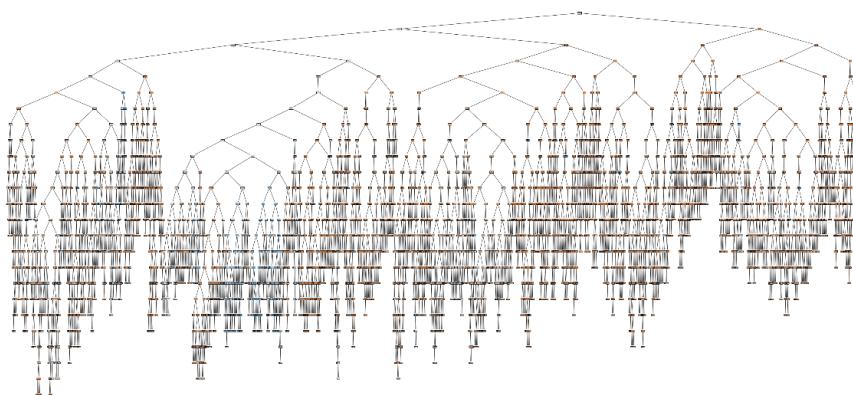
Hình 4.21: Đoạn code của Random Forest Model.

- Tính toán ma trận nhầm lẫn:

```
-----Random Forest-----
Random Forest
Accuracy: 0.7922458762159876
Report:
precision    recall   f1-score   support
          0       0.80      0.90      0.84     22207
          1       0.78      0.62      0.69     13258
accuracy           0.79      0.76      0.77     35465
macro avg         0.79      0.76      0.77     35465
weighted avg      0.79      0.79      0.79     35465
```

Hình 4.22: Kết quả ma trận nhầm lẫn trong lần chạy đầu tiên.

- Mô hình rừng ngẫu nhiên:



Hình 4.23: Mô hình Random Forest trong lần chạy đầu tiên.

- Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

```

-----5. Random Forest-----
print("Time: {}".format(time_rf))
print("Training Accuracy: {}".format(training_score_rf))
print("Test Accuracy: {}".format(test_score_rf))

-----5. Random Forest-----
Time: [4.731429815292358, 6.945342540740967, 4.728205783843994, 5.705215692520142, 4.850972890853882, 4.8217222690582275, 6.20741868019104]
Training Accuracy: [0.798129327742263, 0.7992652656765478, 0.7990960834310159, 0.7985281144638735, 0.7989148167393747, 0.7992773501226571,
0.8001353457964254]
Test Accuracy: [0.7922458762159876, 0.7894543916537431, 0.7906950514591851, 0.7927252220499084, 0.7908924291555054, 0.7892852107711829,
0.7882983222895813]

```

Hình 4.24: Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

6. Logistic Regression Model

6.1. Khái niệm Logistic Regression Model

- Hồi quy logistic là một mô hình phân tích thống kê để dự đoán một kết quả nhị phân (1 hoặc 0, Có hoặc Không, Đúng hoặc Sai) cho một tập hợp các biến độc lập. Có thể coi hồi quy logistic như một trường hợp đặc biệt của hồi quy tuyến tính khi biến kết quả có tính phân loại, trong đó chúng ta đang sử dụng log tỷ lệ cược làm biến phụ thuộc. Nói cách đơn giản, nó dự đoán xác suất xảy ra một sự kiện bằng cách khớp dữ liệu với một hàm logit.

6.2. Ưu điểm

- Hồi quy logistic dễ triển khai, diễn giải hơn nhưng mang lại hiệu quả đào tạo lớn trong một số trường hợp. Cũng vì những lý do này, việc đào tạo một mô hình với thuật toán này không yêu cầu khả năng tính toán cao.
- Phân loại rất nhanh các bản ghi không xác định.
- Độ chính xác tốt cho nhiều tập dữ liệu đơn giản và nó hoạt động tốt khi tập dữ liệu có thể phân tách tuyến tính.
- Nó có thể giải thích các hệ số của mô hình như là các chỉ số về tầm quan trọng của tính năng.
- Thay vì bắt đầu ngay với một mô hình phức tạp, hồi quy logistic đôi khi được sử dụng làm mô hình chuẩn để đo lường hiệu suất, vì nó tương đối nhanh chóng và dễ thực hiện.
- Hồi quy logistic ít có xu hướng overfitting nhưng mô hình vẫn có thể gặp phải tình trạng này nếu tập dữ liệu có high dimensional (một tập dữ liệu trong đó số lượng đối tượng p lớn hơn số lượng quan sát N), có thể tham khảo kỹ thuật Regularization (L1 và L2) để tránh overfitting trong trường hợp này.

6.3. Giải thuật Logistic Regression

- Kiểm định thời gian chạy của thuật toán

```
start= time.time()
lr = LogisticRegression()
lr.fit(x_train,y_train)
end = time.time()
```

Hình 4.25: Đoạn code lưu lại thời gian chạy.

- Chạy thuật toán bằng những dòng lệnh sau

```

print("-----Logistic Regression-----")
start= time.time()
lr = LogisticRegression()
lr.fit(x_train,y_train)
end = time.time()
time_lr.append(end - start)
training_score_lr.append(lr.score(x_train, y_train))
lr_pred = lr.predict(x_test)
lr_score = metrics.accuracy_score(y_test, lr_pred)
test_score_lr.append(lr_score)
print("Logistic Regression")
print("Accuracy:",lr_score)
print("Report:",metrics.classification_report(y_test,lr_pred))
lr_cm = metrics.confusion_matrix(y_test,lr_pred)

```

Hình 4.26: Đoạn code của Logistic Regression Model.

- Tính toán ma trận nhầm lẫn:

```

-----Logistic Regression-----
Logistic Regression
Accuracy: 0.7844353588044551
Report:              precision    recall   f1-score   support
                   0          0.78      0.91      0.84     22207
                   1          0.79      0.58      0.67     13258

accuracy           0.78      0.78      0.78     35465
macro avg         0.78      0.74      0.75     35465
weighted avg      0.78      0.78      0.78     35465

```

Hình 4.27: Kết quả ma trận nhầm lẫn trong lần chạy đầu tiên.

- Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

```

print("-----6. Logistic Regression-----")
print("Time: {}".format(time_lr))
print("Training Accuracy: {}".format(training_score_lr))
print("Test Accuracy: {}".format(test_score_lr))

-----
Time: [0.4991421699523926, 0.6212272644042969, 0.7177331447601318, 0.5271022319793701, 0.5303397178649902, 0.514223575592041, 0.6854310035705566]
Training Accuracy: [0.7823478411233781, 0.7839059346714843, 0.7825524767072302, 0.782274534467137, 0.7821416055395101, 0.7827820811833089, 0.7832292056893573]
Test Accuracy: [0.7844353588044551, 0.7824615818412519, 0.7822078105174115, 0.7846327365007754, 0.7833638798815734, 0.7827435499788524, 0.7807415762018892]

```

Hình 4.28: Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

7. K - Nearest Neighbor Model

7.1. Khái niệm K - Nearest Neighbor Model

- K - Nearest neighbor (KNN) là một thuật toán học máy có giám sát được sử dụng để giải quyết các vấn đề phân loại và hồi quy , KNN có gắng dự đoán đúng lớp cho dữ liệu testing bằng cách tính toán khoảng cách giữa dữ liệu testing và tất cả các điểm training. Sau đó, chọn K số điểm có giá trị phù hợp với dữ liệu testing và tính xác suất dữ liệu testing thuộc các lớp dữ liệu training K và lớp xác suất các nhóm cao nhất sẽ được chọn.

7.2. Ưu điểm

- KNN được gọi là Lazy Learner vì nó không thực hiện bất kỳ khóa training nào khi bạn cung cấp dữ liệu training. Thay vào đó, nó chỉ lưu trữ dữ liệu trong thời gian training và không thực hiện bất kỳ phép tính nào. Nó không xây dựng một mô hình cho đến khi một truy vấn được thực hiện trên tập dữ liệu. Điều này làm cho thuật toán KNN nhanh hơn nhiều so với các thuật toán khác yêu cầu đào tạo và trở nên lý tưởng cho việc khai thác dữ liệu.
- KNN rất dễ thực hiện. Chỉ có hai tham số được yêu cầu để triển khai KNN, tức là giá trị của K và hàm khoảng cách.
- Vì mô hình KNN không yêu cầu dữ liệu được đào tạo trước khi đưa ra dự đoán, dữ liệu mới có thể được thêm vào một cách liền mạch mà độ chính xác của thuật toán sẽ không ảnh hưởng.

7.3. Giải thuật K - Nearest Neighbor

- Kiểm định thời gian chạy của thuật toán

```
start= time.time()
knn = KNeighborsClassifier()
knn.fit(x_train,y_train)
end = time.time()
```

Hình 4.29: Đoạn code lưu lại thời gian chạy.

- Chạy thuật toán bằng những dòng lệnh sau

```

print("-----K-Nearest Neighbor-----")
start= time.time()
knn = KNeighborsClassifier()
knn.fit(x_train,y_train)
end = time.time()
time_knn.append(end - start)
training_score_knn.append(knn.score(x_train, y_train))
knn_pred = knn.predict(x_test)
knn_score = metrics.accuracy_score(y_test, knn_pred)
test_score_knn.append(knn_score)
print("K-Nearest Neighbor")
print("Accuracy:",knn_score)
print("Report:",metrics.classification_report(y_test,knn_pred))
knn_cm = metrics.confusion_matrix(y_test,knn_pred)

```

Hình 4.30: Đoạn code của K - Nearest Neighbor Model.

- Tính toán ma trận nhầm lẫn:

```

-----K-Nearest Neighbor-----
K-Nearest Neighbor
Accuracy: 0.7544339489637671
Report:
precision      recall   f1-score   support
          0       0.77      0.86      0.81     22207
          1       0.71      0.57      0.64     13258

accuracy           0.75
macro avg       0.74      0.72      0.73     35465
weighted avg    0.75      0.75      0.75     35465

```

Hình 4.31: Kết quả ma trận nhầm lẫn trong lần chạy đầu tiên.

- Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

```

print("-----7. K-Nearest Neighbor-----")
print("Time: {}".format(time_knn))
print("Training Accuracy: {}".format(training_score_knn))
print("Test Accuracy: {}".format(test_score_knn))

-----7. K-Nearest Neighbor-----
Time: [0.0030007362365722656, 0.005918979644775391, 0.005106449127197266, 0.003999948501586914, 0.003318309783935547, 0.01562643051147461,
0.00500178337097168]
Training Accuracy: [0.7595678602071274, 0.7626614784111371, 0.7567038464791966, 0.756305059757586, 0.7688245459269374, 0.7534893838140929,
0.777960387185653]
Test Accuracy: [0.7544339489637671, 0.7533342732271253, 0.7493021288594389, 0.7524601720005639, 0.7612011842661779, 0.7477513041026363,
0.7716903989849146]

```

Hình 4.32 : Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

8. Neural Network Model

8.1. Khái niệm Neural Network Model

- Short Term: Mạng thần kinh là một mô hình đơn giản hóa cách bộ não con người xử lý thông tin. Nó hoạt động bằng cách mô phỏng một số lượng lớn các đơn vị xử lý được kết nối với nhau giống như các phiên bản trừu tượng của tế bào thần kinh.
- Long Term: Mạng thần kinh là mạng sử dụng các mô hình toán học phức tạp để xử lý thông tin. Chúng dựa trên mô hình hoạt động của các tế bào thần kinh và khớp thần kinh trong não của con người. Tương tự như bộ não con người, một mạng lưới thần kinh kết nối các nút đơn giản, còn được gọi là tế bào thần kinh hoặc đơn vị. Và một tập hợp các nút như vậy tạo thành một mạng lưới các nút, do đó có tên là "mạng thần kinh". Tương tự như bộ não con người, trong mạng nơ-ron, một loạt các thuật toán được sử dụng để xác định và nhận dạng các mối quan hệ trong tập dữ liệu. Mạng nơron được thiết kế để thích ứng với các kích bản đầu vào động; với kết quả đó, mạng cung cấp các kết quả tốt nhất có thể mà không cần xử lý thêm ở kết quả đầu ra.

8.2. Ưu điểm

- Mô hình này lưu trữ thông tin trên toàn bộ mạng, nếu một phần thông tin biến mất ở một ô nào đó thì nó cũng không ngăn được sự hoạt động của mạng.
- Việc đưa ra kết quả đầu ra không bị ảnh hưởng bởi sự hỏng hóc của một hoặc nhiều ô của neural network làm cho các mạng có khả năng chịu lỗi tốt hơn
- Có khả năng training machine do neural network học hỏi từ các sự kiện và đưa ra quyết định thông qua bình luận về các sự kiện tương tự.
- Có khả năng xử lý song song do nó có sức mạnh về số lượng khiến chúng có khả năng thực hiện nhiều hơn một chức năng cùng một lúc.

8.3. Giải thuật Neural Network

- Kiểm định thời gian chạy của thuật toán

```
start= time.time()
nn = MLPClassifier(random_state=0, activation="logistic", solver="sgd")
nn.fit(x_train,y_train)
end = time.time()
```

Hình 4.34: Đoạn code lưu lại thời gian chạy.

- Chạy thuật toán bằng những dòng lệnh sau

```
print("-----Neural Network-----")
start= time.time()
nn = MLPClassifier(random_state=0, activation="logistic", solver="sgd")
nn.fit(x_train,y_train)
end = time.time()
time_nn.append(end - start)
training_score_nn.append(nn.score(x_train, y_train))
nn_pred = nn.predict(x_test)
nn_score = metrics.accuracy_score(y_test, nn_pred)
test_score_nn.append(nn_score)
print("Neural Network")
print("Accuracy:",nn_score)
print("Report:",metrics.classification_report(y_test,nn_pred))
nn_cm = metrics.confusion_matrix(y_test,nn_pred)
```

Hình 4.35: Đoạn code của Neural Network Model.

- Tính toán ma trận nhầm lẫn:

```
-----Neural Network-----
Neural Network
Accuracy: 0.7831665021852531
Report:
precision    recall   f1-score   support
          0       0.78      0.90      0.84     22207
          1       0.78      0.58      0.67     13258

accuracy                           0.78      35465
macro avg       0.78      0.74      0.75      35465
weighted avg    0.78      0.78      0.77      35465
```

Hình 4.36: Kết quả ma trận nhầm lẫn trong lần chạy đầu tiên.

- Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

```
print("-----8. Neural Network-----")
print("Time: {}".format(time_nn))
print("Training Accuracy: {}".format(training_score_nn))
print("Test Accuracy: {}".format(test_score_nn))

-----
Time: [34.92758584022522, 32.73545980453491, 33.0911078453064, 38.42025709152222, 33.752549171447754, 36.77770185470581, 45.37167429924011]
Training Accuracy: [0.7811748498507571, 0.7827458278449807, 0.7825162233689019, 0.7816944810334618, 0.7813561165423982, 0.7828183345216372,
0.7831083612282631]
Test Accuracy: [0.7831665021852531, 0.7811927252220499, 0.7815592838009305, 0.7845481460594953, 0.7827435499788524, 0.7820104328210912,
0.7804032144367686]
```

Hình 4.37: Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

9. Bagging Classifier Model

9.1. Khái niệm Bagging Classifier Model

- Bagging Classifier là một loại phương pháp học máy tổng hợp kết hợp các kết quả đầu ra từ nhiều người học để cải thiện hiệu suất. Mô hình này hoạt động bằng cách chia nhỏ tập huấn luyện thành các tập con và chạy chúng thông qua các mô hình học máy khác nhau, sau đó kết hợp các dự đoán của chúng khi chúng quay lại với nhau để tạo ra dự đoán tổng thể cho từng trường hợp trong dữ liệu gốc.

9.2. Ưu điểm

- Dễ triển khai do các thư viện Python như scikit-learning đưa ra các mô-đun có sẵn mà bạn có thể tận dụng để tối ưu hóa mô hình của mình.
- Giảm phương sai trong một thuật toán learning. Điều này đặc biệt hữu ích với tập dữ liệu có high dimensional, trong đó các giá trị bị thiếu có thể dẫn đến phương sai cao hơn, khiến nó dễ bị overfitting và ngăn chặn accurate generalization cho các tập dữ liệu mới.

9.3. Giải thuật Bagging Classifier

- Kiểm định thời gian chạy của thuật toán

```
start= time.time()
bc = BaggingClassifier(random_state=0)
bc.fit(x_train,y_train)
end = time.time()
```

Hình 4.38: Đoạn code lưu lại thời gian chạy.

- Chạy thuật toán bằng những dòng lệnh sau

```

print("-----Bagging Classifier-----")
start= time.time()
bc = BaggingClassifier(random_state=0)
bc.fit(x_train,y_train)
end = time.time()
time_bc.append(end - start)
training_score_bc.append(bc.score(x_train, y_train))
bc_pred = bc.predict(x_test)
bc_score = metrics.accuracy_score(y_test, bc_pred)
test_score_bc.append(bc_score)
print("Bagging Classifier")
print("Accuracy:",bc_score)
print("Report:",metrics.classification_report(y_test,bc_pred))
bc_cm = metrics.confusion_matrix(y_test,bc_pred)

```

Hình 4.39: Đoạn code của Bagging Classifier Model.

- Tính toán ma trận nhầm lẫn:

```

-----Bagging Classifier-----
Bagging Classifier
Accuracy: 0.7905258705766248
Report:
          precision    recall   f1-score   support
          0         0.80     0.89     0.84    22207
          1         0.77     0.62     0.69    13258

      accuracy           0.79
macro avg            0.79     0.76     0.77    35465
weighted avg          0.79     0.79     0.79    35465

```

Hình 4.40: Kết quả ma trận nhầm lẫn trong lần chạy đầu tiên.

- Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

```

print("-----9. Bagging Classifier-----")
print("Time: {}".format(time_bc))
print("Training Accuracy: {}".format(training_score_bc))
print("Test Accuracy: {}".format(test_score_bc))

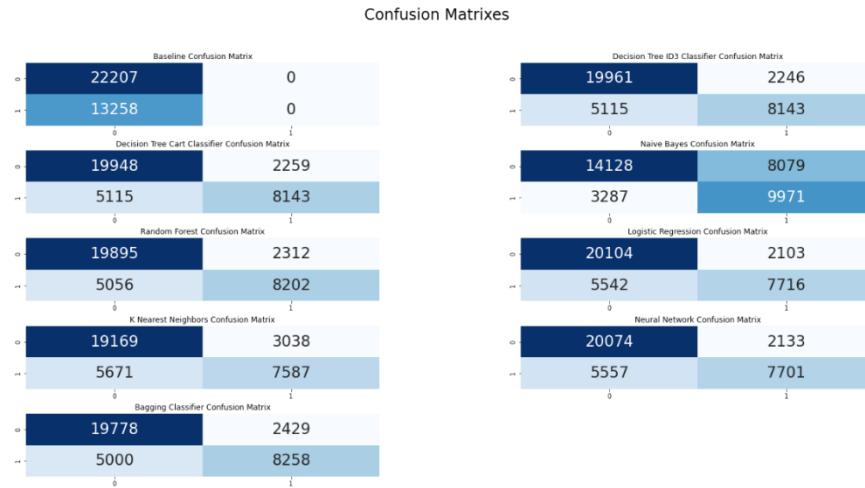
-----
-----9. Bagging Classifier-----
Time: [1.6422724723815918, 1.507767677307129, 1.7803282737731934, 1.609457015991211, 1.8360247611999512, 1.9092953205108643, 1.8356618881225586]
Training Accuracy: [0.79765880343439958, 0.7986489589249677, 0.7986610433710771, 0.798129327742263, 0.7985522833560924, 0.7989027322932654, 0.798857403535909]
Test Accuracy: [0.7905258705766248, 0.7888058649372621, 0.7891160298886226, 0.7921330889609474, 0.790638657831665, 0.7888904553785422, 0.7876497955731002]

```

Hình 4.41: Kết quả thời gian chạy, độ chính xác của mô hình huấn luyện và độ chính xác của mô hình kiểm tra.

CHƯƠNG 5: PHÂN TÍCH ĐÁNH GIÁ CÁC THUẬT TOÁN VÀ DỰ BÁO

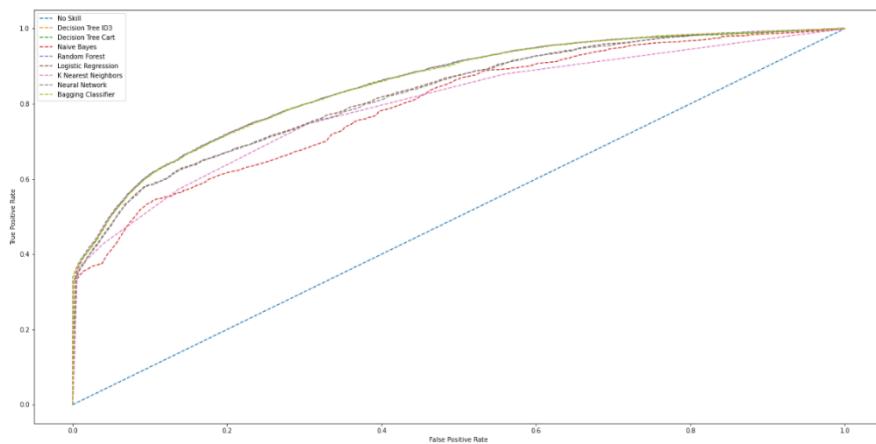
1. Ma trận nhầm lẫn



Hình 5.1: Ma trận nhầm lẫn trong lần chạy đầu tiên.

- Nhận xét: Tỷ lệ TPR và FNR cao hơn TNR và FPR cho thấy mô hình hợp lệ.

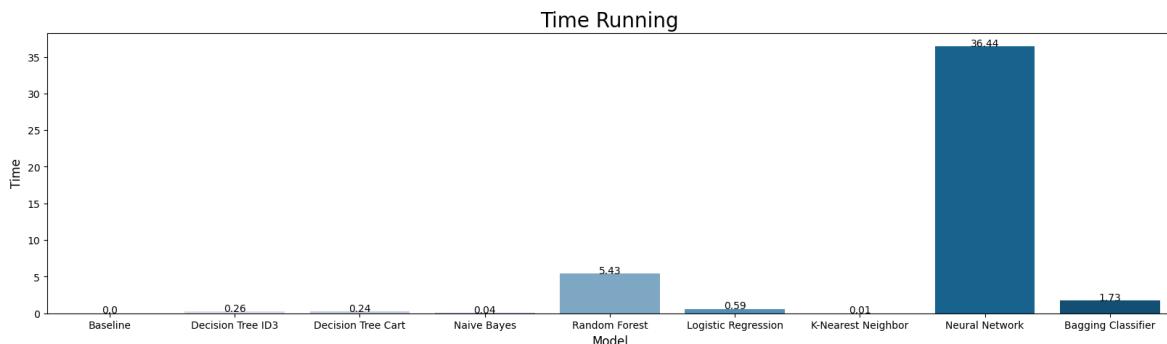
2. Đánh giá tỷ lệ TPR (True positive rate) trên tỷ lệ FPR (False positive rate)



Hình 5.2: Sơ đồ ROC.

- Nhận xét: các mô hình Decision Tree ID3, Decision Tree Cart, Random Forest và Bagging Classifier có độ chính xác cao và tương đương nhau.

3. Đánh giá về thời gian chạy thuật toán

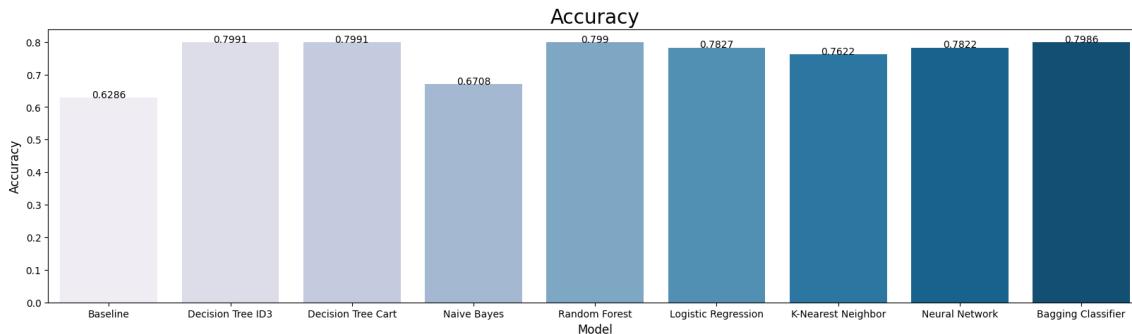


Hình 5.3: Đồ thị biểu diễn thời gian chạy của 9 thuật toán.

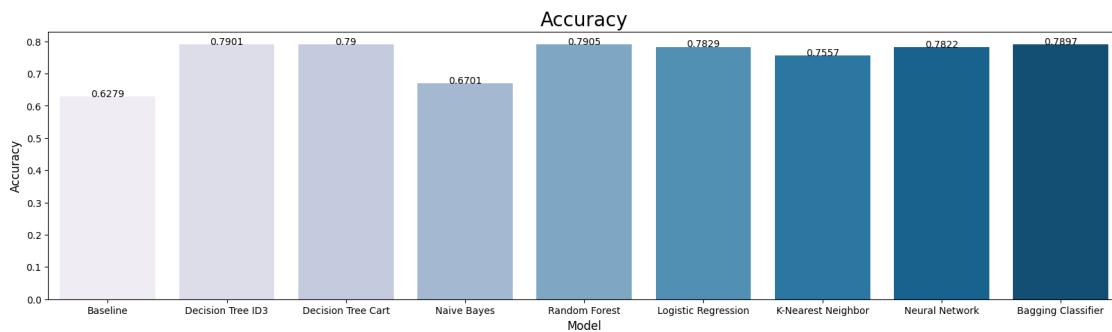
- Nhận xét:

- 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 về độ chính xác của thuật toán



Hình 5.4: Đồ thị biểu diễn độ chính xác của mô hình huấn luyện.

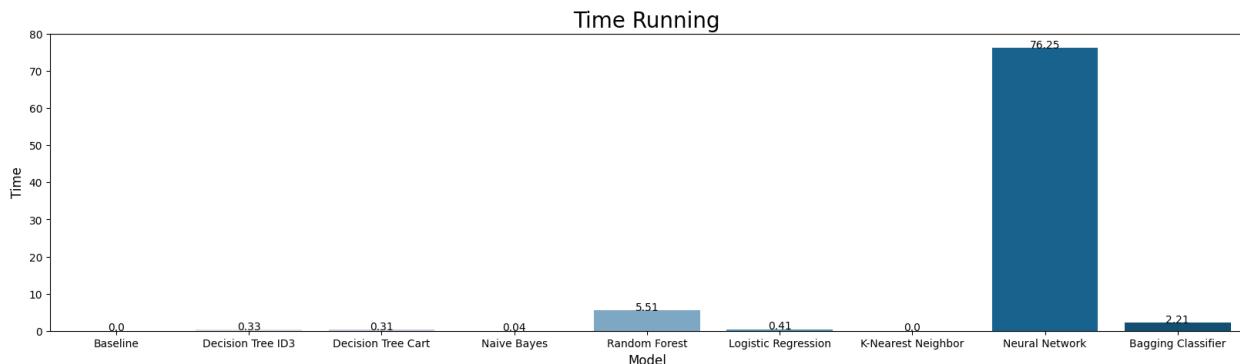


Hình 5.5: Đồ thị biểu diễn độ chính xác của mô hình kiểm định.

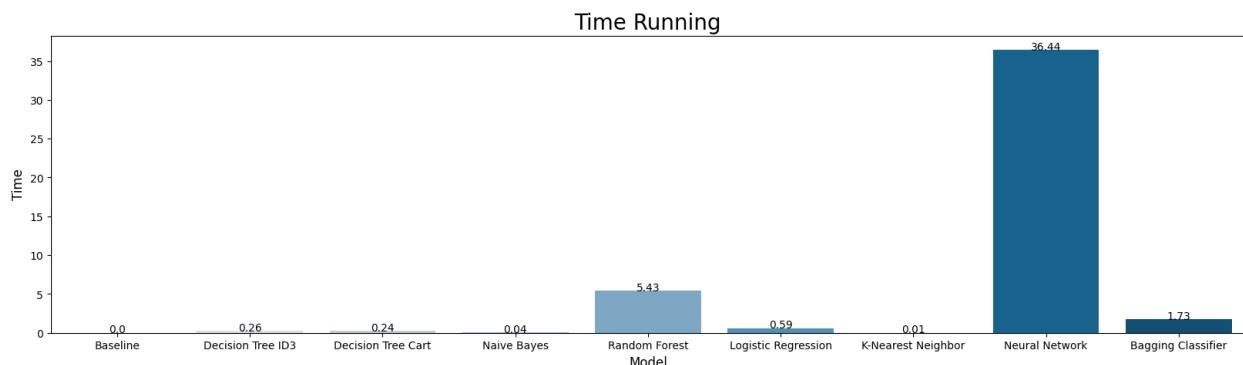
- Nhận xét: Độ chính xác của mô hình Decision Tree trong kiểm định gần như tương đương với mô hình khi huấn luyện.

5. Đánh giá thuật toán

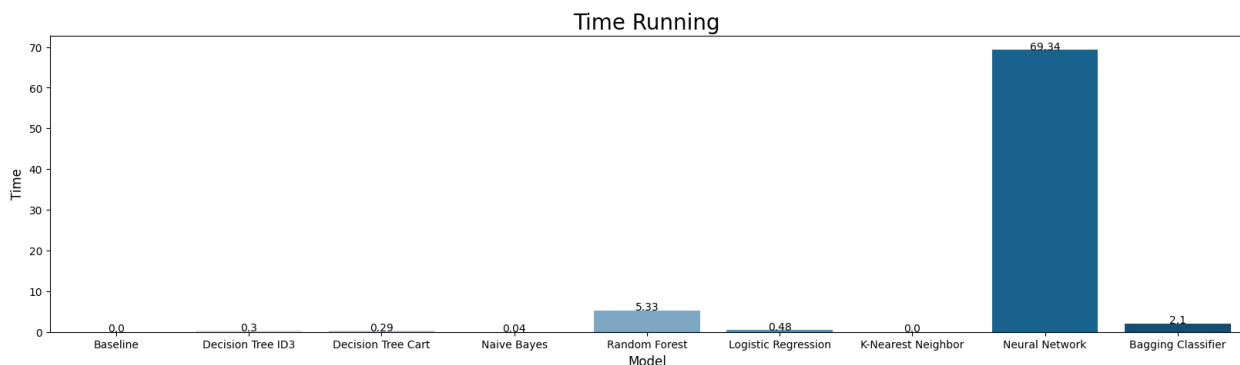
5.1. So sánh thời gian chạy



Hình 5.6: Thời gian chạy khi loại bỏ mối tương quan lớn hơn 0.6.

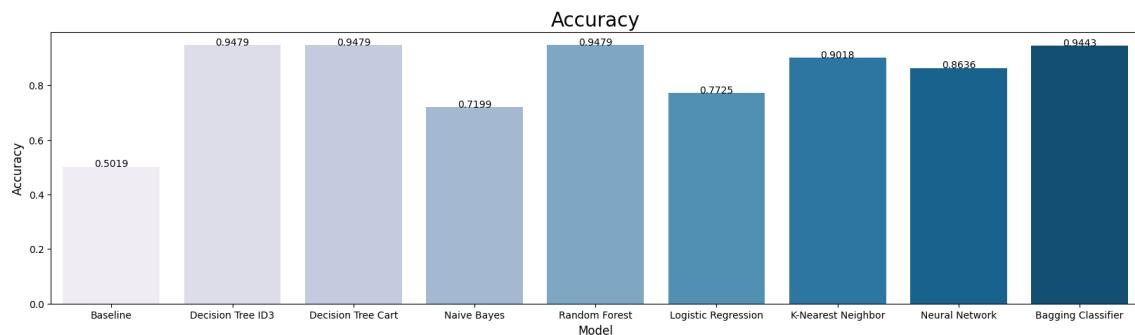


Hình 5.7: Thời gian chạy khi loại bỏ mối tương quan lớn hơn 0.7 và có chuẩn hoá.

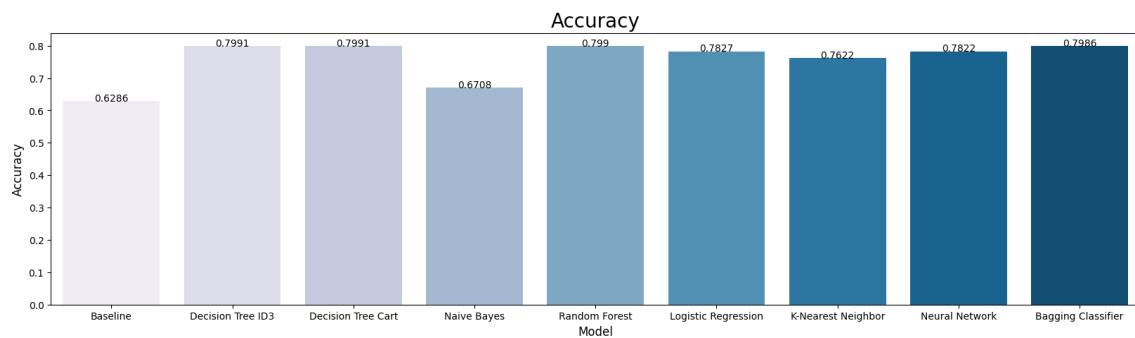


Hình 5.8: Thời gian chạy khi loại bỏ mối tương quan lớn hơn 0.7.

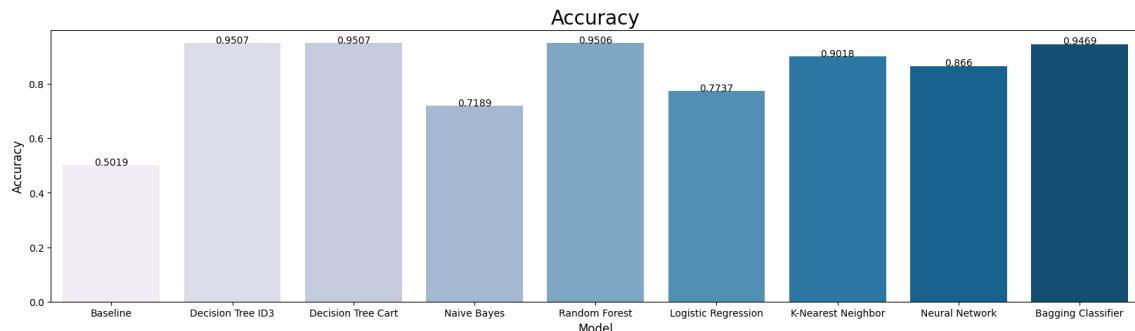
5.2. So sánh độ chính xác của tập huấn luyện



Hình 5.9: Độ chính xác của tập huấn luyện chạy khi loại bỏ mỗi tương quan lớn hơn 0.6.

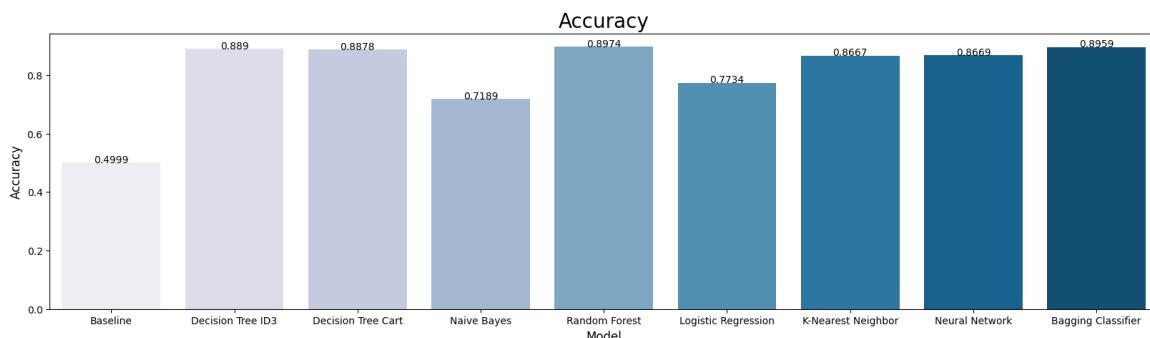


Hình 5.10: Độ chính xác của tập huấn luyện chạy khi loại bỏ mỗi tương quan lớn hơn 0.7 và có chuẩn hóa.

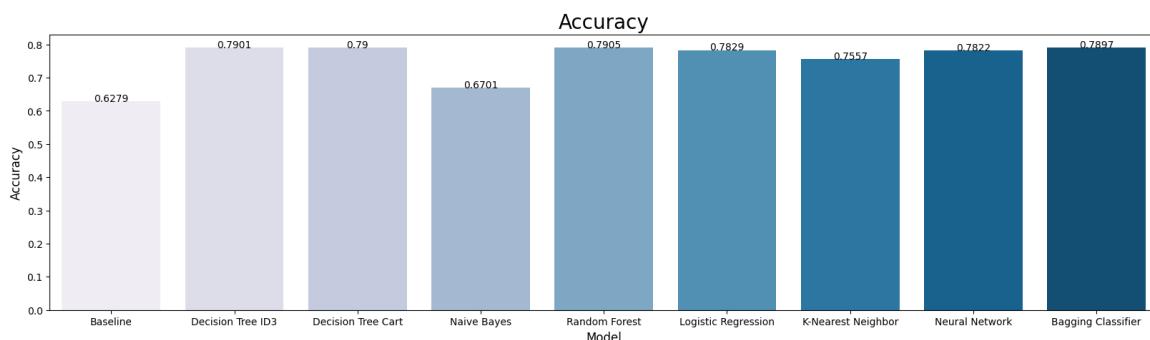


Hình 5.11: Độ chính xác của tập huấn luyện chạy khi loại bỏ mỗi tương quan lớn hơn 0.7.

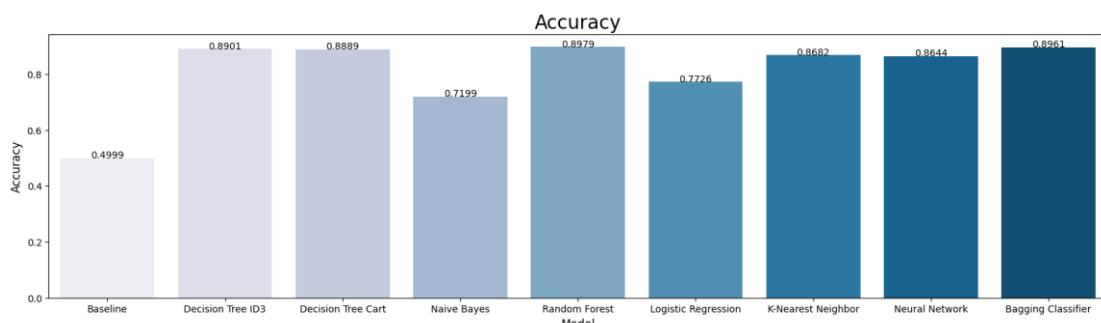
5.3. So sánh độ chính xác của tập thử nghiệm



Hình 5.12: Độ chính xác của tập huấn luyện chạy khi loại bỏ mối tương quan lớn hơn 0.6.



Hình 5.13: Độ chính xác của tập huấn luyện chạy khi loại bỏ mối tương quan lớn hơn 0.7 và có chuẩn hóa.



Hình 5.14: Độ chính xác của tập huấn luyện chạy khi loại bỏ mối tương quan lớn hơn 0.7.

5.4. So sánh các thuật toán

5.4.1. Decision Tree

❖ *Ưu điểm đối với tập dữ liệu:*

- Ngoài không yêu cầu chuẩn hóa dữ liệu, Decision Tree cũng không yêu cầu xử lý dữ liệu thì độ chính xác của mô hình càng giảm tuy nhiên thời gian chạy sẽ nhanh hơn.
- Tuy nhiên, nhìn chung, thuật toán đạt hiệu quả về mặt thời gian và cả độ chính xác.

❖ *Khuyết điểm đối với tập dữ liệu:*

- Có nhiều nhánh không cần thiết nên cần phải tiến hành xử lý để rút ngắn thời gian hơn nữa.

5.4.2. Naïve Bayes

❖ *Ưu điểm đối với tập dữ liệu.*

- Thuật toán hoạt động rất nhanh và dễ dàng dự đoán lớp của tập dữ liệu. Do đặc trưng và nhãn đã được chuyển hóa thành dạng số nên ít tốn thời gian chạy. Hơn thế nữa, Naïve Bayes sử dụng ít tài nguyên nên quá trình triển khai nhẹ và nhanh chóng.
- Đối với các trường hợp chưa được mã hóa, Naïve Bayes có độ chính xác cao hơn, vì mô hình này không bị ảnh hưởng nghiêm trọng bởi các thuật toán khác.

❖ *Khuyết điểm đối với tập dữ liệu.*

- Thuật toán hoạt động tốt trên các đặc trưng độc lập nên độ chính xác của Naïve Bayes không quá cao trong tập dữ liệu vì các thuộc tính có mối tương quan với nhau.

5.4.3. Random Forest

❖ *Ưu điểm đối với tập dữ liệu.*

- Mô hình không gặp vấn đề Overfitting.
- Độ chính xác trong tập kiểm định và thử nghiệm đều rất cao

❖ *Khuyết điểm đối với tập dữ liệu.*

- Thời gian chạy của Random Forest khá chậm vì tập dữ liệu quá lớn cần vẽ nhiều cây.
- Cần xử lý dữ liệu trước khi thực thi thì thời gian chạy càng nhanh.

5.4.4. Logistic Regression

- ❖ *Ưu điểm đối với tập dữ liệu.*
 - Dữ liệu càng chuẩn hóa và xử lý thì thời gian chạy càng nhanh và độ chính xác càng cao.
- ❖ *Khuyết điểm đối với tập dữ liệu.*
 - Khi càng đưa vào các biến độc lập không quan trọng và không liên quan đến mô hình, chất lượng mô hình sẽ càng giảm xuống.
 - Logistic Regression khá nhạy với các ngoại lệ sẽ đưa ra mô hình không có độ chính xác cao.
 - Các biến đầu vào phải độc lập với nhau để chúng không bị đa cộng tác.

5.4.5. K – Nearest Neighbor

- ❖ *Ưu điểm đối với tập dữ liệu.*
 - Thời gian chạy nhanh nhờ cơ chế Lazy Learner.
 - Độ chính xác sau khi được chuẩn hóa đạt giá trị cao nhất.
- ❖ *Khuyết điểm đối với tập dữ liệu.*
 - 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.

5.4.6. Neural Network

- ❖ *Ưu điểm đối với tập dữ liệu.*
 - Độ chính xác khi chuẩn hóa dữ liệu xấp xỉ nhau.
- ❖ *Khuyết điểm đối với tập dữ liệu.*
 - 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.

5.4.7. Bagging Classifier

- ❖ *Ưu điểm đối với tập dữ liệu.*
 - Thời gian chạy tại các trường hợp nhanh.
 - Độ chính xác của Bagging tương đối cao như các thuật toán khác như Decision Tree và Random Forest.
- ❖ *Khuyết điểm đối với tập dữ liệu.*
 - Độ chênh lệch giữa độ chính xác theo thử nghiệm và theo huấn luyện cao.
 - Tốn kém về mặt tính toán và có thể không phù hợp trong các trường hợp sử dụng khác nhau do Bagging Classifier xây dựng nhiều mô hình.

6. Đánh giá làm việc

- *Ưu điểm:* Biết cách tìm hiểu, khai thác dữ liệu (tiền xử lý, phân lớp bằng các thuật toán đã học) đánh giá kết quả, làm việc nhóm, xây dựng mô hình dự báo hỗ trợ cho công tác điều tra tội phạm.
- *Nhược điểm:* Vì thời gian trong một học kỳ nên nhóm chưa kịp tìm ra cách để tối ưu độ chính xác cao nhất, chưa am hiểu rành mạch về ngôn ngữ Python.
- *Hướng phát triển:* Sẽ nghiên cứu các thuật toán để giảm cột trong việc tiền xử lý dữ liệu để chọn ra cột nào làm cho thuật toán có độ chính xác cao nhất.

BẢNG PHÂN CHIA CÔNG VIỆC

	Đặng Vũ Phương Uyên 19520345	Nguyễn Thị Thu Phương 19522066	Nguyễn Ngọc Thiện 19522264
Chọn dữ liệu, thảo luận đánh giá dữ liệu.	x	x	x
Mô tả dữ liệu, mô tả bài toán.			x
Trục quan hóa dữ liệu trước khi tiền xử lý.		x	
Mô tả và làm sạch dữ liệu.	x		
Tích hợp dữ liệu, biến đổi dữ liệu và thu giảm dữ liệu.		x	
Rời rạc hóa dữ liệu, Sampling và thu giảm số chiều dữ liệu.			x
Thực hiện thuật toán Baseline, Decision Tree ID3 và Decision Tree Cart.			x
Thực hiện thuật toán Naive Bayes, Random Forest và Logistic Regression.		x	
Thực hiện thuật toán K – Nearest Neighbor, Neural Network và Bagging Classifier.	x		
Đánh giá thời gian, độ chính xác của thuật toán.	x		
Đưa ra ưu điểm, nhược điểm và hướng phát triển thuật toán.	x	x	x
Xây dựng tài liệu, viết báo cáo và tìm tài liệu tham khảo.	x	x	x

TÀI LIỆU THAM KHẢO

- [1] Slide bài giảng lý thuyết của Ths.Hà Lê Hoài Trung.
- [2] Tài liệu hướng dẫn thực hành của Ths.Hà Lê Hoài Trung.
- [3] Scikit – learn : <https://scikit-learn.org/stable/>
- [3] Giải thuật chia giỏ : <https://pbpython.com/pandas-qcut-cut.html>