BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC BÀ RỊA – VŨNG TÀU

KHOA CÔNG NGHỆ KỸ THUẬT - NÔNG NGHIỆP CÔNG NGHỆ CAO

BARIA VUNGTAU UNIVERSITY

CAP SAINT JACQUES

BÁO CÁO MÔN HỌC

ỨNG DỤNG TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI

SỬ DỤNG THUẬT TOÁN LOGISTIC REGRESSION K – NEAREST NEIGHBOR DECISION TREE ĐỂ PHÂN TÍCH DỮ LIỆU

Sinh viên thực hiện: Vi Hoàng Gia

MSSV: 18033885

Lớp: DH19CT

Ngành: Công nghệ thông tin

Vũng Tàu, Tháng 12 2022

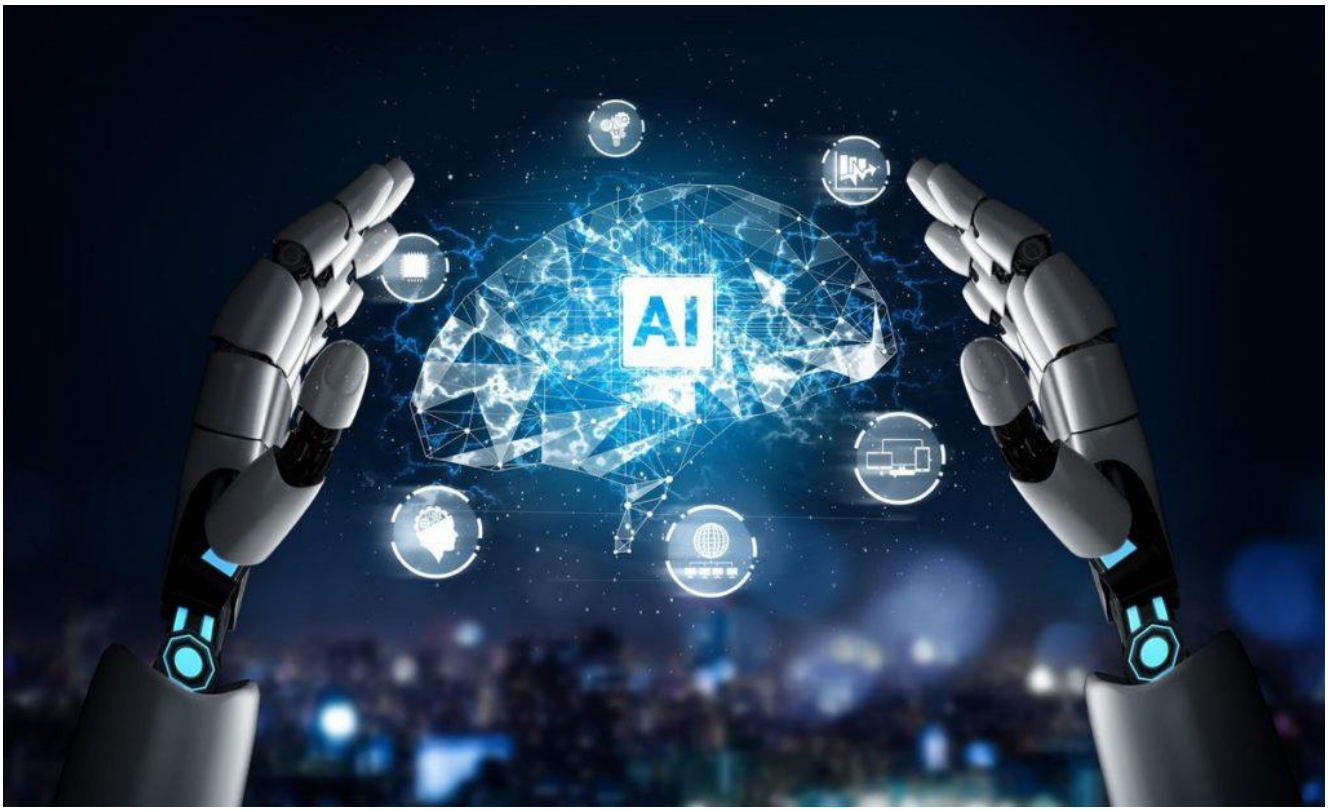
MỤC LỤC

Nhận xét của giảng viên

CHƯƠNG I: Giới thiệu

I. Máy học và những ứng dụng của máy học

Thuật ngữ học máy có lẽ không còn quá xa lạ với con người ngày nay bởi chúng đã được sử dụng trên phạm vi toàn thế giới. Hiện tại, Machine Learning đã đạt được rất nhiều thành tựu to lớn và trở thành hướng nghiên cứu chính trong AI (trí tuệ nhân tạo). Vậy học máy là gì? Đừng bỏ lỡ bài viết sau, chúng mình sẽ giúp bạn tìm hiểu những thông tin liên quan đến Machine Learning.



1. Học máy là gì?

Học máy (machine learning) là khả năng của chương trình máy tính sử dụng kinh nghiệm, quan sát, hoặc dữ liệu trong quá khứ để cải thiện công việc của mình trong tương lai thay vì chỉ thực hiện theo đúng các quy tắc đã được lập trình sẵn. Chẳng hạn, máy tính có thể học cách dự đoán dựa trên các ví dụ, hay học cách tạo ra các hành vi phù hợp dựa trên quan sát trong quá khứ.

Ví dụ. Xét một số ví dụ sau. Ví dụ thứ nhất là học cách đánh cờ. Chương trình có thể quan sát các ván cờ cùng với kết quả (thắng hay thua) để cải thiện khả năng chơi cờ và tăng số ván thắng trong tương lai. Trong trường hợp này, kinh nghiệm là các ván cờ trong quá khứ (có thể là ván cờ chương trình tự chơi với chính mình), được sử dụng để học cách làm tốt hơn công việc chơi cờ với tiêu chí đánh giá là số ván thắng.

Ví dụ thứ hai là học nhận dạng các ký tự. Chương trình được cung cấp dữ liệu dưới dạng ảnh chụp các ký tự (chữ cái) cùng mã UNICODE của ký tự đó. Sau khi học, chương trình cần có khả năng nhận dạng các ảnh chụp ký tự mới, tức là xác định được mã UNICODE của các ảnh mới chụp ký tự đã được học.

Tương tự quá trình học thông thường, một hệ thống học máy cần có khả năng ghi nhớ, thích nghi, và đặc biệt là tổng quát hóa. Tổng quát hóa là khả năng của hệ thống học máy ra quyết định chính xác trong các trường hợp mới, chưa gặp, dựa trên kinh nghiệm học được từ dữ liệu hoặc các quan sát trước đó.

Lý do cần tới học máy

Học máy là một nhánh nghiên cứu rất quan trọng của trí tuệ nhân tạo với khá nhiều ứng dụng thành công trong thực tế. Hiện nay, học máy là một trong những lĩnh vực phát triển mạnh nhất của trí tuệ nhân tạo. Có một số lý do giải thích cho sự cần thiết và phát triển của học máy:

- Thứ nhất, rất khó xây dựng hệ thống thông minh có thể thực hiện các công việc liên quan đến trí tuệ như thị giác máy, xử lý ngôn ngữ tự nhiên mà không sử dụng tới kinh nghiệm và quá trình học. Thông thường, khi viết chương trình, cần có thuật toán rõ ràng để chuyển đổi đầu vào thành đầu ra. Tuy nhiên, trong nhiều bài toán, rất khó để xây dựng được thuật toán như vậy. Như trong ví dụ về nhận dạng chữ ở trên, người bình thường có khả năng nhận dạng các chữ rất tốt nhưng rất khó để giải thích vì sao từ đầu vào là ảnh lại kết luận được đây là ký tự cụ thể nào. Học máy cho phép tìm ra giải pháp cho những trường hợp như vậy dựa trên dữ liệu, chẳng hạn bằng cách tìm ra điểm chung và riêng giữa rất nhiều ảnh chụp các ký tự.
- Thứ hai, nhiều ứng dụng đòi hỏi chương trình máy tính phải có khả năng thích nghi. Ví dụ, hành vi mua sắm của khách hàng có thể thay đổi theo thời điểm cụ thể trong ngày, trong năm, hoặc theo tuổi tác. Việc xây dựng thuật toán cố định cho những ứng dụng cần thích nghi và thay đổi là không phù hợp. Học máy mang lại khả năng thích nghi nhờ phân tích dữ liệu thu thập được.
- Thứ ba, việc tìm được chuyên gia và thu thập được tri thức cần thiết cho việc thiết kế thuật toán để giải quyết các vấn đề tương đối khó, trong khi dữ liệu ngày càng nhiều và có thể thu thập dễ dàng hơn. Khả năng lưu trữ và tính toán của máy tính cũng ngày càng tăng, cho phép thực hiện thuật toán học máy trên dữ liệu có kích thước lớn.
- Cuối cùng, bản thân khả năng học là một hoạt động trí tuệ quan trọng của con người, do vậy học tự động hay học máy luôn thu hút được sự quan tâm khi xây dựng hệ thống thông minh.

2. Ứng dụng của học máy

Có rất nhiều ứng dụng thực tế khác nhau của học máy. Hai lĩnh vực ứng dụng lớn nhất của học máy là khai phá dữ liệu (data mining) và nhận dạng mẫu (pattern recognition).

Khai phá dữ liệu là ứng dụng kỹ thuật học máy vào các cơ sở dữ liệu hoặc các tập dữ liệu lớn để phát hiện quy luật hay tri thức trong dữ liệu đó hoặc để dự đoán các thông tin quan tâm trong tương lai. Ví dụ, từ tập hợp hóa đơn bán hàng có thể phát hiện ra quy luật "những người mua bánh mì thường mua bơ".

Nhận dạng mẫu là ứng dụng các kỹ thuật học máy để phát hiện các mẫu có tính quy luật trong dữ liệu, thường là dữ liệu hình ảnh, âm thanh. Bài toán nhận dạng mẫu cụ thể thường là xác định nhãn cho đầu vào cụ thể, ví dụ cho ảnh chụp mặt người, cần xác định đó là ai.

Cần lưu ý, khai phá dữ liệu và nhận dạng mẫu có nhiều điểm trùng nhau cả trong phạm vi

nghiên cứu và ứng dụng. Điểm khác nhau chủ yếu liên quan tới lĩnh vực ứng dụng và kỹ thuật sử dụng, theo đó khai phá dữ liệu liên quan tới dữ liệu thương mại trong khi nhận dạng mẫu liên quan nhiều tới dữ liệu âm thanh, hình ảnh và được dùng nhiều trong kỹ thuật.

Ứng dụng cụ thể

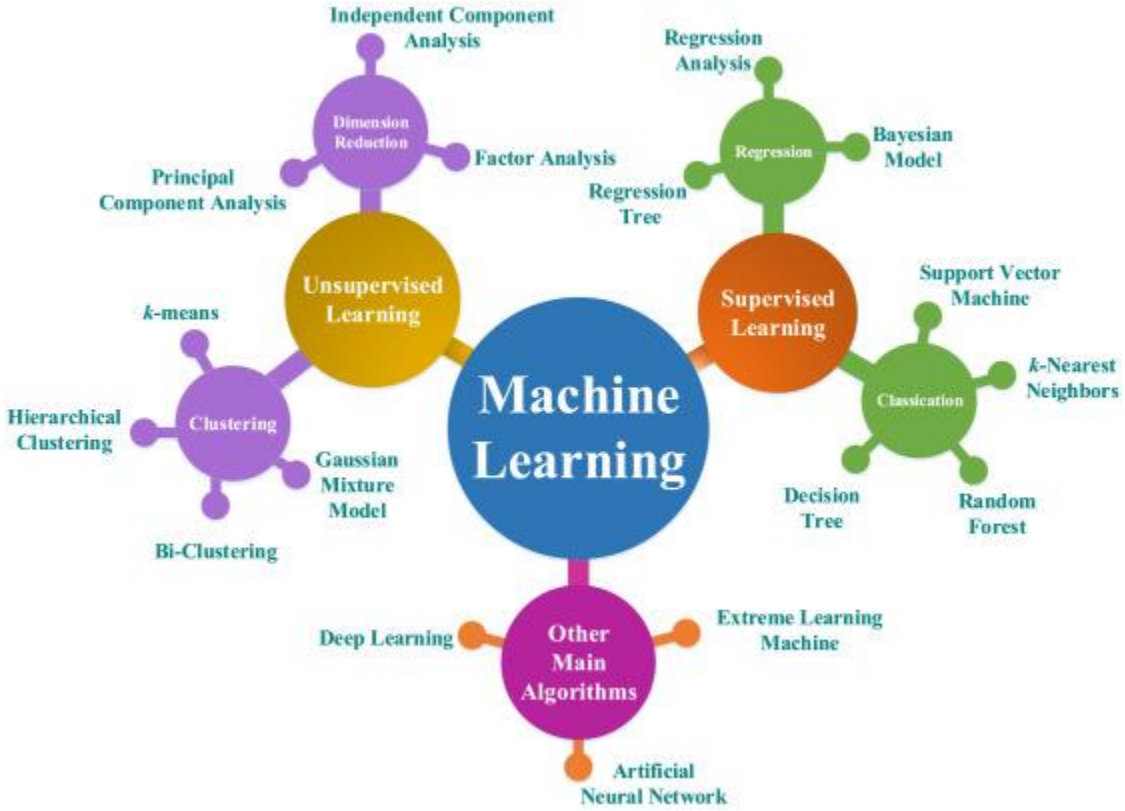
Sau đây là một số ví dụ ứng dụng cụ thể của học máy:

- Nhận dạng ký tự: phân loại hình chụp ký tự thành các loại, mỗi loại ứng với một ký tự tương ứng.
- Phát hiện và nhận dạng mặt người: phát hiện vùng có chứa mặt người trong ảnh, xác định đó là mặt người nào trong số những người đã có ảnh trước đó, tức là phân chia ảnh thành những loại tương ứng với những người khác nhau.
- Lọc thư rác, phân loại văn bản: dựa trên nội dung thư điện tử, chia thư thành loại "thư rác" hay "thư bình thường"; hoặc phân chia tin tức thành các thể loại khác nhau như "xã hội", "kinh tế", "thể thao".v.v
- Dịch tự động: dựa trên dữ liệu huấn luyện dưới dạng các văn bản song ngữ, hệ thống dịch tự động học cách dịch từ ngôn ngữ này sang ngôn ngữ khác. Hệ thống dịch tự động tiêu biểu dạng này là Google Translate.
- Chẩn đoán y tế: học cách dự đoán người bệnh có mắc hay không mắc một số bệnh nào đó dựa trên triệu chứng quan sát được.
- Phân loại khách hàng và dự đoán sở thích: sắp xếp khách hàng vào một số loại, từ đây dự đoán sở thích tiêu dùng của khách hàng.
- Dự đoán chỉ số thí trường: căn cứ giá trị một số tham số hiện thời và trong lịch sử, đưa ra dự đoán, chẳng hạn dự đoán giá chứng khoán, giá vàng.v.v.
- Các hệ khuyến nghị, hay hệ tư vấn lựa chọn: cung cấp một danh sách ngắn các loại hàng hóa, phim, video, tin tức v.v. mà người dùng nhiều khả năng quan tâm. Ví dụ ứng dụng loại này là phần khuyến nghị trên Youtube hay trên trang mua bán trực tuyến Amazon.
- Ứng dụng lái xe tự động: dựa trên các mẫu học chứa thông tin về các tình huống trên đường, hệ thống học máy cho phép tự ra quyết định điều khiển xe, và do vậy không cần người lái. Hiện Google đã có kế hoạch thương mại hóa xe ôtô tự động lái như vậy.

3. Các dạng học máy

Khi thiết kế và xây dựng hệ thống học máy cần quan tâm tới những yếu tố sau.

- Thứ nhất, kinh nghiệm hoặc dữ liệu cho học máy được cho dưới dạng nào?
- Thứ hai, lựa chọn biểu diễn cho hàm đích ra sao? Hàm đích có thể biểu diễn dưới dạng hàm đại số thông thường nhưng cũng có thể biểu diễn dưới những dạng khác như dạng cây, dạng mạng nơ ron, công thức xác suất .v.v.



Việc sử dụng những dạng kinh nghiệm và dạng biểu diễn khác nhau dẫn tới những dạng học máy khác nhau. Có ba dạng học máy chính như sau:

*** Học có giám sát** (supervised learning). Là dạng học máy trong đó cho trước tập dữ liệu huấn luyện dưới dạng các ví dụ cùng với giá trị đầu ra hay giá trị đích. Dựa trên dữ liệu huấn luyện, thuật toán học cần xây dựng mô hình hay hàm đích để dự đoán giá trị đầu ra (giá trị đích) cho các trường hợp mới.

- Nếu giá trị đầu ra là rời rạc thì học có giám sát được gọi là phân loại hay phân lớp (classification).
- Nếu đầu ra nhận giá trị liên tục, tức đầu ra là số thực, thì học có giám sát được gọi là hồi quy (regression). Trong phần tiếp theo, ta sẽ xem xét chi tiết hơn về học có giám sát

*** Học không giám sát** (un-supervised learning). Là dạng học máy trong đó các ví dụ được cung cấp nhưng không có giá trị đầu ra hay giá trị đích.

- Thay vì xác định giá trị đích, thuật toán học máy dựa trên độ tương tự giữa các ví dụ để xếp chúng thành những nhóm, mỗi nhóm gồm các ví dụ tương tự nhau. Hình thức học không giám sát như vậy gọi là phân cụm (clustering). Ví dụ, chỉ bằng cách quan sát hoặc đo chiều cao của mọi người, dần dần ta học được khái niệm "người cao" và "người thấp", và có thể xếp mọi người vào hai cụm tương ứng.
- Ngoài phân cụm, một dạng học không giám sát phổ biến khác là phát hiện luật kết hợp (association rule). Luật kết hợp có dạng P(A | B), cho thấy xác suất hai tính chất A và B xuất hiện cùng với Ví dụ, qua phân tích dữ liệu mua hàng ở siêu

thị, ta có luật P(Bơ | Bánh mỳ) =80%, có nghĩa là 80% những người mua bánh mỳ cũng mua bơ.

*** Học tăng cường** (reinforcement learning). Đối với dạng học này, kinh nghiệm không được cho trực tiếp dưới dạng đầu vào/đầu ra cho mỗi trạng thái hoặc mỗi hành động. Thay vào đó, hệ thống nhận được một giá trị khuyến khích (reward) là kết quả cho một chuỗi hành động nào đó. Thuật toán cần học cách hành động để cực đại hóa giá trị khuyến khích. Ví dụ của học khuyến khích là học đánh cờ, trong đó hệ thống không được chỉ dẫn nước đi nào là hợp lý cho từng tình huống mà chỉ biết kết quả toàn ván cờ. Như vậy, các chỉ dẫn về nước đi được cho một cách gián tiến và có độ trễ dưới dạng giá trị thưởng. Nước đi tốt là nước đi nằm trong một chuỗi các nước đi dẫn tới kết quả thắng toàn bộ ván cờ.

Trong các dạng học máy, học có giám sát là dạng phổ biến, có nhiều thuật toán liên quan và nhiều ứng dụng nhất.

CHƯƠNG II: Thuật toán phân loại và phân tích dữ liệu

I. Thuật toán phân loại

- Logistic Regression

Hồi quy logistic là một kỹ thuật phân tích dữ liệu sử dụng toán học để tìm ra mối quan hệ giữa hai yếu tố dữ liệu. Sau đó, kỹ thuật này sử dụng mối quan hệ đã tìm được để dự đoán giá trị của những yếu tố đó dựa trên yếu tố còn lại. Dự đoán thường cho ra một số kết quả hữu hạn, như có hoặc không.

- K-Nearest Neighbor

K-nearest neighbor là một trong những thuật toán supervised-learning đơn giản nhất (mà hiệu quả trong một vài trường hợp) trong Machine Learning. Khi training, thuật toán này không học một điều gì từ dữ liệu training (đây cũng là lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới. K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán Supervised learning là Classification và Regression. KNN còn được gọi là một thuật toán Instance-based hay Memory-based learning.

- Decision Tree

Cây quyết định (Decision Tree) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật. Các thuộc tính của đối tượngncó thể thuộc các kiểu dữ liệu khác nhau như Nhị phân (Binary) , Định danh (Nominal), Thứ tự (Ordinal), Số lượng (Quantitative) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal.

II. Dữ liệu

1. Bài toán kinh doanh

- Dự đoán liệu khách hàng có chấp thuận phiếu giảm giá hay không là một bài toán khó và chúng ta không thể chỉ giới thiệu nó cho tất cả mọi người vì các chi phí liên quan. Vì vậy, trong bài toán này, chúng ta sẽ dự đoán xem khách hàng sẽ chấp thuận hay từ chối phiếu giảm giá được cung cấp dựa trên hồ sợ và lịch sử khách hàng.

2. Yêu cầu

- Mục tiêu của bài toán là dự đoán xem khách hàng sẽ chấp nhận hay từ chối phiếu giảm giá một địa điểm cụ thể dựa trên các thuộc tính. Nếu khách hàng chấp nhận phiếu giảm giá sẽ dán nhãn Y=1 và nếu khách hàng từ chối phiếu giảm giá sẽ được đánh giá Y=0.

3. Dữ liệu

- Dữ liệu này được thu thập thông qua một cuộc khảo sát trên Amazon Mechanical Turk được xuất bản trên trang web UCI vào năm 2017. Cuộc khảo sát mô tả các tình huống lái xe khác nhau bao gồm điểm đến người dùng, thời gian, thời tiết, hành khách, thuộc tính phiếu giảm giá,… Sau đó hỏi xem người dùng có chấp thuận phiếu giảm giá không.

  Tệp dữ liệu: in-vehicle-coupon-recommendation.csv

  Bộ dữ liệu bao gồm hơn 12684 dòng dữ liệu và 26 đặc trưng

Giải thích về các thuộc tính/ đặc trưng của bộ dữ liệu:

Thuộc tính người dùng:

- Gender: Female, Male
- Age:21, 46, 26, 31, 41, 50plus, 36, below21
- Marital Status: Unmarried partner, Single, Married partner, Divorced, Widowed
- has_Children: 1: has children, 0: No children
- Education: Some college — no degree, Bachelors degree, Associates degree, High School Graduate, Graduate degree (Masters or Doctorate), Some High School
- Occupation: nghề nghiệp của người dùng (Unemployed, Architecture & Engineering, Student,Education&Training&Library, Healthcare Support,Healthcare Practitioners & Technical, Sales & Related, Management,Arts Design Entertainment Sports & Media, Computer & Mathematical,Life Physical Social Science, Personal Care & Service, Community & Social Services, Office & Administrative Support, Construction & Extraction, Legal, Retired, Installation Maintenance & Repair, Transportation & Material Moving, Business & Financial, Protective Service, Food Preparation & Serving Related, Production Occupations, Building & Grounds Cleaning & Maintenance, Farming Fishing & Forestry)
- Income: thu nhập của người dùng (Less than $12500,$12500 — $24999,$25000 — $37499,$37500 — $49999,$50000 — $62499,$62500 — $74999,$75000 — $87499,$87500 — $99999,$100000 or More)

- Car : Mô tả phương tiện do người dùng điều kiển (Scooter and motorcycle, crossover, Mazda5) (99% of values are missing in this feature)
- Bar: người dùng đến bar bao nhiêu lần mỗi tháng (never, less1, 1~3, 4~8, gt8, nan)
- CoffeeHouse: người dùng đến quán cà phê bao nhiêu lần mỗi tháng (never, less1, 1~3, 4~8, gt8, nan)
- CarryAway: người dùng nhận đồ ăn mang về bao nhiêu lần mỗi tháng (never, less1, 1~3, 4~8, gt8, nan)
- RestaurantLessThan20: bao nhiêu lần người dùng đến nhà hàng với chi phí trung bình cho mỗi người dưới 20$ mỗi tháng? (never, less1, 1~3, 4~8, gt8, nan)
- Restaurant20To50: người dùng đến nhà hàng bao nhiêu lần với chi phí trung bình cho mỗi ngườ là 20 – 50$ mỗi tháng? (never, less1, 1~3, 4~8, gt8, nan)

bình cho mỗi ngườ là 20 – 50$ mỗi tháng? (never, less1, 1~3, 4~8, gt8, nan)

Thuộc tính ngữ cảnh:

- Destination: điểm đến của người dùng (No Urgent Place, Home, Work)
- Passenger: khách hàng trên xe là ai (Alone, Friend(s), Kid(s), Partner)
- Weather: thời tiết khi người dùng đang lái xe (Sunny, Rainy, Snowy)
- Temperature: nhiệt độ tính bằng độ F khi người dùng lái xe (55, 80, 30)
- Time: thời gian khi người dùng lái xe (2PM, 10AM, 6PM, 7AM, 10PM)
- toCoupon_GEQ5min: khoảng cách lái xe đến nhà hàng/quán cà phê/ quán bar để sử dụng phiếu giảm giá lớn hơn 5 phút (0,1)
- toCoupon_GEQ15min: khoảng cách lái xe đến nhà hàng/quán cà phê/ quán bar để sử dụng phiếu giảm giá lớn hơn 15 phút (0,1)
- toCoupon_GEQ25min: khoảng cách lái xe đến nhà hàng/quán cà phê/ quán bar để sử dụng phiếu giảm giá lớn hơn 25 phút (0,1)
- direction_same: liệu nhà hàng/ quán cà phê/ quán bar có cùng hướng với điểm đến hiện tại của người dùng hay không (0,1)
- direction_opp: liệu nhà hàng/ quán cà phê/ quán bar có ở hướng ngược lại với điểm đến hiện tại của người dùng hay không (0,1)

Thuộc tính phiếu giảm giá

- Coupon: loại phiếu giảm giá do công ty cung cấp (Restaurant(<$20), Coffee House, Carry out & Take away, Bar, Restaurant($20-$50))

Ở đây, Restaurant(<$20) là mức trả trung bình cho mỗi người dùng dưới $20, nhà hàng không quá đắt. Restaurant($20-$50) có nghĩa là mức trả trung bình cho mỗi người dùng là từ $20 đến $50, nhà hàng hơi đắt tiền

- Expiration: Phiếu giảm giá hết hạn sau 1 ngày hoặc sau 2h (1d, 2h)

Thuộc tính đích:

- Y: phiếu giảm giá được chấp thuận hay từ chối, 1: được chấp thuận , 0: bị từ chối

III. Phân tích dữ liệu

Link source code github : https://github.com/ViHoangGia/Tri_Tue_Nhan_Tao_Cuoi_Ky

1. Thư viện

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme(style="whitegrid")

from sklearn.metrics import log_loss,roc_auc_score

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV

from prettytable import PrettyTable
from tqdm import tqdm
```

- Thư viện pandas để đọc, ghi, xử lý file dữ liệu
- Thư viện Numpy để làm việc với các ma trận và mảng
- Thư viện Matplotlib dùng để vẽ đồ thị
- Thư viện seaborn để tạo ra các hình ảnh trực quan đẹp mắt. Nó được coi như 1 phần mở rộng của matplotlib
- Thư viện sklearn là 1 thư viện mạnh mẽ tích hợp rất nhiều thuật toán dành cho Machine Learning
- Prettytable dùng để dễ đàng hiển thị dữ liệu ở dạng bảng ở định dạng bảng ASCII
- Tqdm là 1 tiện ích của python nó giúp hiển thị các vòng lặp dưới dạng 1 giao diện thông minh

2. Tải dữ liệu

Để tải dữ liệu, chúng ta chỉ cần tệp in-vehicle-coupon-recommendation.csv và tải tệp này bằng khung dữ liệu pandas.

```
data = pd.read_csv("in-vehicle-coupon-recommendation.csv")
data.head()
```

✓ 0.1s Python

	destination	passanger	weather	temperature	time	coupon	expiration	gender	age	maritalStatus	...	CoffeeHouse	CarryAway	RestaurantLessThan20	Restau
0	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d	Female	21	Unmarried partner	...	never	NaN	4~8	
1	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House	2h	Female	21	Unmarried partner	...	never	NaN	4~8	
2	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away	2h	Female	21	Unmarried partner	...	never	NaN	4~8	
3	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	2h	Female	21	Unmarried partner	...	never	NaN	4~8	
4	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	1d	Female	21	Unmarried partner	...	never	NaN	4~8	

5 rows × 26 columns

```
data.info()
```

✓ 0.5s

```
Output exceeds the size limit. Open the full output data in a text editor
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12684 entries, 0 to 12683
Data columns (total 26 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   destination           12684 non-null  object
 1   passanger             12684 non-null  object
 2   weather               12684 non-null  object
 3   temperature           12684 non-null  int64
 4   time                  12684 non-null  object
 5   coupon                12684 non-null  object
 6   expiration            12684 non-null  object
 7   gender                12684 non-null  object
 8   age                   12684 non-null  object
 9   maritalStatus         12684 non-null  object
 10  has_children          12684 non-null  int64
 11  education             12684 non-null  object
 12  occupation            12684 non-null  object
 13  income                12684 non-null  object
 14  car                   108 non-null    object
 15  Bar                   12577 non-null  object
```

Bộ dữ liệu này có 12684 dòng dữ liệu và 26 đặc trưng

3. Phân bổ coupon

```
Y_counts = data.groupby('Y').Y.count()
print('Phiếu giảm giá được chấp nhận:',Y_counts[1],round(Y_counts[1]/data.shape[0]*100,3),'%')
print('Phiếu giảm giá bị từ chối:',Y_counts[0],round(Y_counts[0]/data.shape[0]*100,3),'%')
```

✓ 0.2s

```
Phiếu giảm giá được chấp nhận: 7210 56.843 %
Phiếu giảm giá bị từ chối: 5474 43.157 %
```

Tập dữ liệu này được cân bằng một phần với nhãn loại chấp nhận là khoảng 57% và loại từ chối là khoảng 43%

4. Loại bỏ trùng lặp

```
# Xóa lặp
duplicate = data[data.duplicated(keep = 'last')]
data = data.drop_duplicates()
print("Tệp dữ liệu sau khi xóa bỏ lặp",data.shape)
```

✓ 0.6s

```
Tệp dữ liệu sau khi xóa bỏ lặp (12610, 26)
```

5. Kiểm tra các dữ liệu còn thiếu trong tệp dữ liệu

```
#Giá trị bị thiếu
print('Có bất kỳ giá trị nào bị thiếu hay không?',data.isnull().values.any())
missing_percentage = data.isnull().sum()*100/len(data)
missing_value_df = pd.DataFrame({'missing_count': data.isnull().sum(),'missing_percentage': missing_percentage})
missing_value_df[missing_value_df.missing_count != 0]
```

✓ 0.9s

```
Có bất kỳ giá trị nào bị thiếu hay không? True
```

	missing_count	missing_percentage
car	12502	99.143537
Bar	107	0.848533
CoffeeHouse	217	1.720856
CarryAway	150	1.189532
RestaurantLessThan20	129	1.022998
Restaurant20To50	189	1.498810

Chúng ta có thể thấy rằng có tổng 6 thuộc tính giá trị bị thiếu, từ tính năng ‘car’ có 99% giá trị bị thiếu nên chúng ta cần bỏ đặc trưng này vì ngay cả sau khi dự đoán các giá trị bị thiếu, đặc trưng này ít quan trọng hơn và nó ít có khả năng dự đoán hơn.

Các đặc trưng khác như ‘Bar’, ‘CoffeeHouse’, ‘CarryAway’, ‘RestaurantLessThan20’, ‘Restaurant20To50’ có khoảng 1% giá trị bị thiếu, vì vậy chúng tôi cần điền các giá trị bị thiếu này bằng 1 số giá trị khác, vì vậy chúng tôi sẽ sử dụng mode imputation.

6. Điền các giá trị bị thiếu bằng cách sử dụng mode imputation

   Mode imputation thay thế các giá trị còn thiếu của một đối tượng phân loại bằng giá trị phổ biến nhất của đối tượng đó.

```
data = data.drop(['car'], axis=1)
```
✓ 0.2s

```
# điền các dữ liệu bị thiếu bằng cách sử dụng mode imputation
data['Bar']=data['Bar'].fillna(data['Bar'].value_counts().index[0])
data['CoffeeHouse']=data['CoffeeHouse'].fillna(data['CoffeeHouse'].value_counts().index[0])
data['CarryAway']=data['CarryAway'].fillna(data['CarryAway'].value_counts().index[0])
data['RestaurantLessThan20']=data['RestaurantLessThan20'].fillna(data['RestaurantLessThan20'].value_counts().index[0])
data['Restaurant20To50']=data['Restaurant20To50'].fillna(data['Restaurant20To50'].value_counts().index[0])
```
✓ 0.4s

7. Mối tương quan giữa các đặc trưng

```
data.corr()
```
✓ 0.3s

```
C:\Users\vihoa\AppData\Local\Temp\ipykernel_1412\2627137660.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecate
version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  data.corr()
```

	temperature	has_children	toCoupon_GEQ5min	toCoupon_GEQ15min	toCoupon_GEQ25min	direction_same	direction_opp	Y
temperature	1.000000	-0.018599	NaN	-0.157089	-0.227165	0.097972	-0.097972	0.059393
has_children	-0.018599	1.000000	NaN	0.079434	-0.010773	-0.032353	0.032353	-0.045056
toCoupon_GEQ5min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
toCoupon_GEQ15min	-0.157089	0.079434	NaN	1.000000	0.321260	-0.302066	0.302066	-0.082693
toCoupon_GEQ25min	-0.227165	-0.010773	NaN	0.321260	1.000000	-0.189900	0.189900	-0.108139
direction_same	0.097972	-0.032353	NaN	-0.302066	-0.189900	1.000000	-1.000000	0.014932
direction_opp	-0.097972	0.032353	NaN	0.302066	0.189900	-1.000000	1.000000	-0.014932
Y	0.059393	-0.045056	NaN	-0.082693	-0.108139	0.014932	-0.014932	1.000000

   Từ ma trận tương quan, chúng ta có thể thấy rằng đặc trưng ‘direction_same’ tương quan hoàn hảo với ‘direction_opp’, cả hai đều có cùng giá trị tương quan với thuộc tính đích. Vì vậy không cần 2 tính năng tương quan hoàn hảo để chúng tôi có thể loại bỏ một trong số chúng.

   Đặc trưng ‘toCoupon_GEQ5min’ không tương quan với thuộc tính đích vì đặc trưng này có cùng giá trị ‘1’ cho tất cả dòng dữ liệu, có nghĩa tất cả các điểm ăn uống đều các người dùng ít nhất 5 phút.

   Vì vậy, cần bỏ cả hai đặc tính ‘direction_opp’ và ‘toCoupon_GEQ5min’ vì đặc tính này ít quan trọng hơn và sẽ không giúp ích nhiều cho việc dự đoán.

```
data = data.drop(['direction_opp','toCoupon_GEQ5min'], axis=1)
data.shape
```
✓ 0.2s

```
(12610, 23)
```

8. Phân tích đơn biến

Vì tất cả các thuộc tính đều có tính phân loại, nên chúng ta sẽ xem xét các phân phối kín đáo của các biến khác nhau với biến mục tiêu.

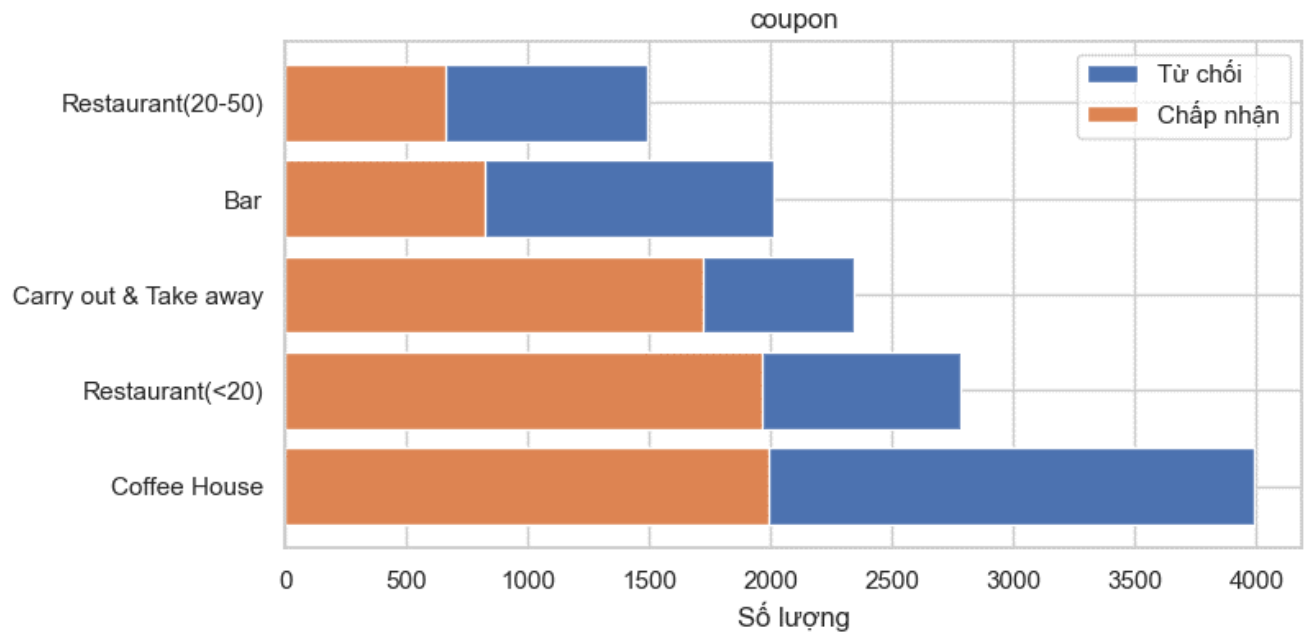
```
def univariate_analysis(column_name):
    df_EDA = pd.DataFrame(sorted(list(data[column_name].unique())),columns=
    [column_name])
    df_EDA['Tổng số lượng'] = list(data.groupby(column_name).Y.count())
    df_EDA['Chấp nhận'] = list(data[data.Y==1].groupby(column_name).Y.count())
    df_EDA['Từ chối'] = list(data[data.Y==0].groupby(column_name).Y.count())
    df_EDA['Phần trăm chấp nhận'] = round(df_EDA['Chấp nhận']/df_EDA['Tổng số
    lượng']*100,3)
    df_EDA['Phần trăm từ chối'] = round(df_EDA['Từ chối']/df_EDA['Tổng số
    lượng']*100,3)
    df_EDA = df_EDA.sort_values(by='Tổng số lượng', ascending=False)

    fig = plt.subplots(figsize =(8, 4))
    plt.barh(df_EDA[column_name],df_EDA['Tổng số lượng'])
    plt.barh(df_EDA[column_name],df_EDA['Chấp nhận'])
    plt.legend(labels=['Từ chối','Chấp nhận'])
    plt.xlabel('Số lượng')
    plt.title(column_name)
    plt.show()

    return df_EDA
```

✓ 0.2s Python

1. Coupon:

```
univariate_analysis('coupon')
```

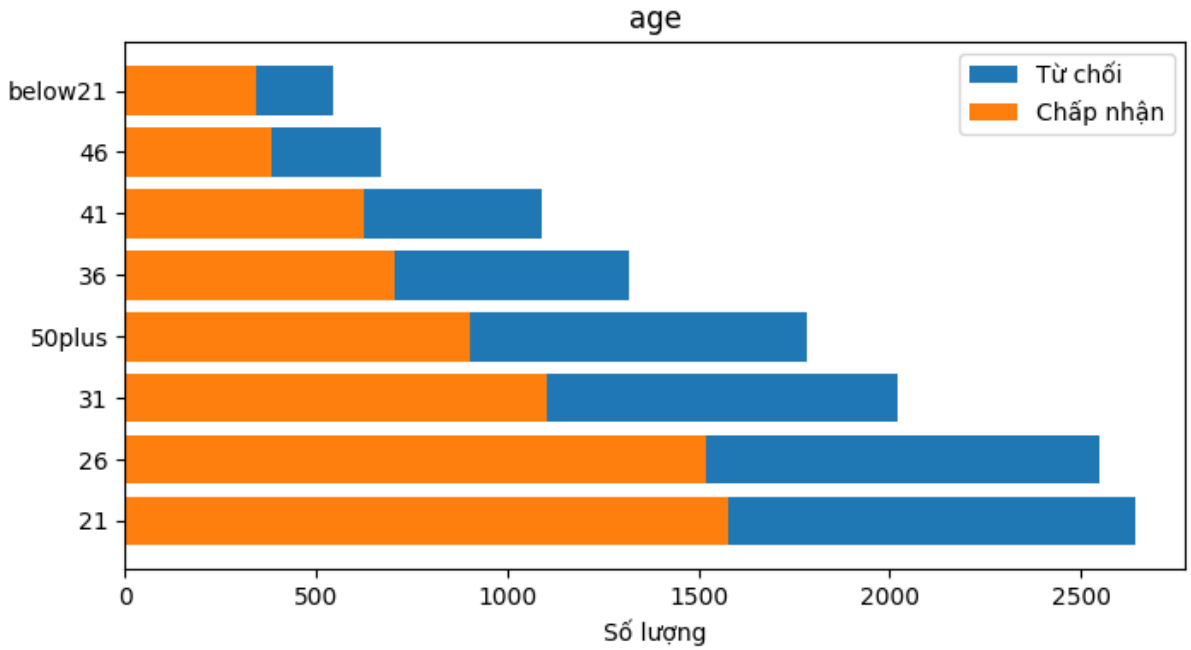
✓ 0.2s



	coupon	Tổng số lượng	Chấp nhận	Từ chối	Phần trăm chấp nhận	Phần trăm từ chối
2	Coffee House	3989	1989	2000	49.862	50.138
4	Restaurant(<20)	2779	1967	812	70.781	29.219
1	Carry out & Take away	2344	1720	624	73.379	26.621
0	Bar	2010	824	1186	40.995	59.005
3	Restaurant(20-50)	1488	657	831	44.153	55.847

- Chúng ta có thể thấy rằng các phiếu giảm giá được cung cấp nhiều nhất là dành cho Coffee House. Ngoài ra, phiếu giảm giá được nhiều người chấp nhận là phiếu giảm giá 'Carry out & Take away' với 73% và 'Restaurant(<20)' là 70%. Phiếu giảm giá có thỉ lệ chấp thuận thấp nhất là 'Bar'

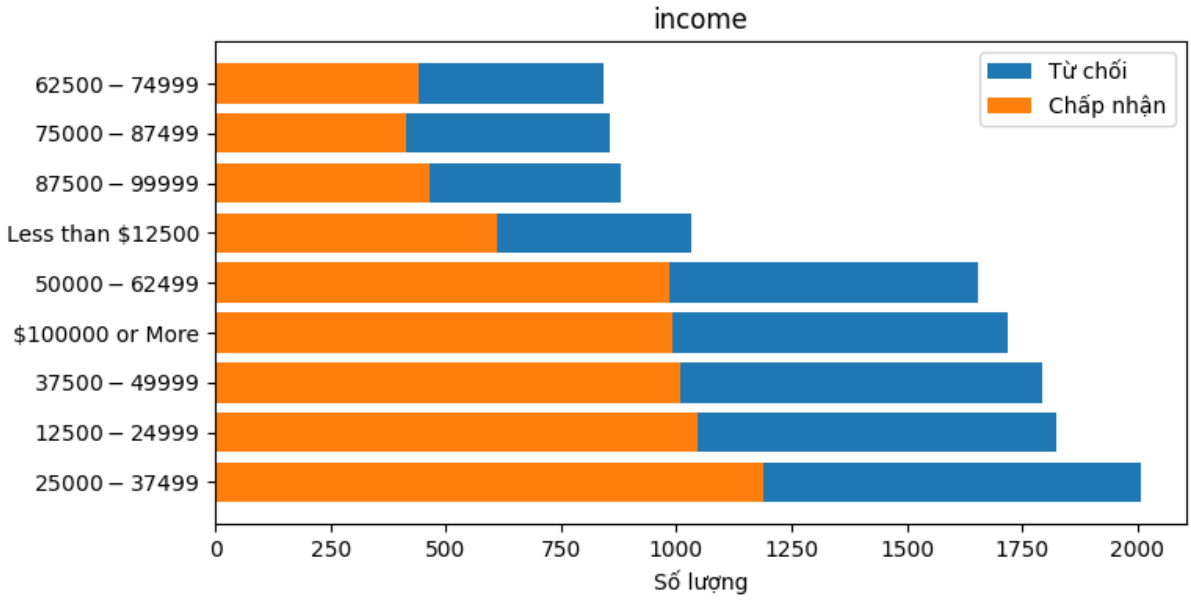
2. Age:



	age	Tổng số lượng	Chấp nhận	Từ chối	Phần trăm chấp nhận	Phần trăm từ chối
0	21	2642	1579	1063	59.765	40.235
1	26	2548	1517	1031	59.537	40.463
2	31	2019	1102	917	54.581	45.419
6	50plus	1781	903	878	50.702	49.298
3	36	1317	705	612	53.531	46.469
4	41	1089	623	466	57.208	42.792
5	46	670	384	286	57.313	42.687
7	below21	544	344	200	63.235	36.765

- Chúng ta có thể thấy rằng hầu hết người dùng trong dữ liệu này có độ tuổi từ 21 đến 31 tuổi. Ngoài ra, người dùng có độ tuổi dưới 21 có tỷ lệ chấp nhận phiếu giảm giá cao nhất. Ngoài ra, người dùng trên 50 tuổi có tỷ lệ từ chối phiếu giảm giá cao nhất

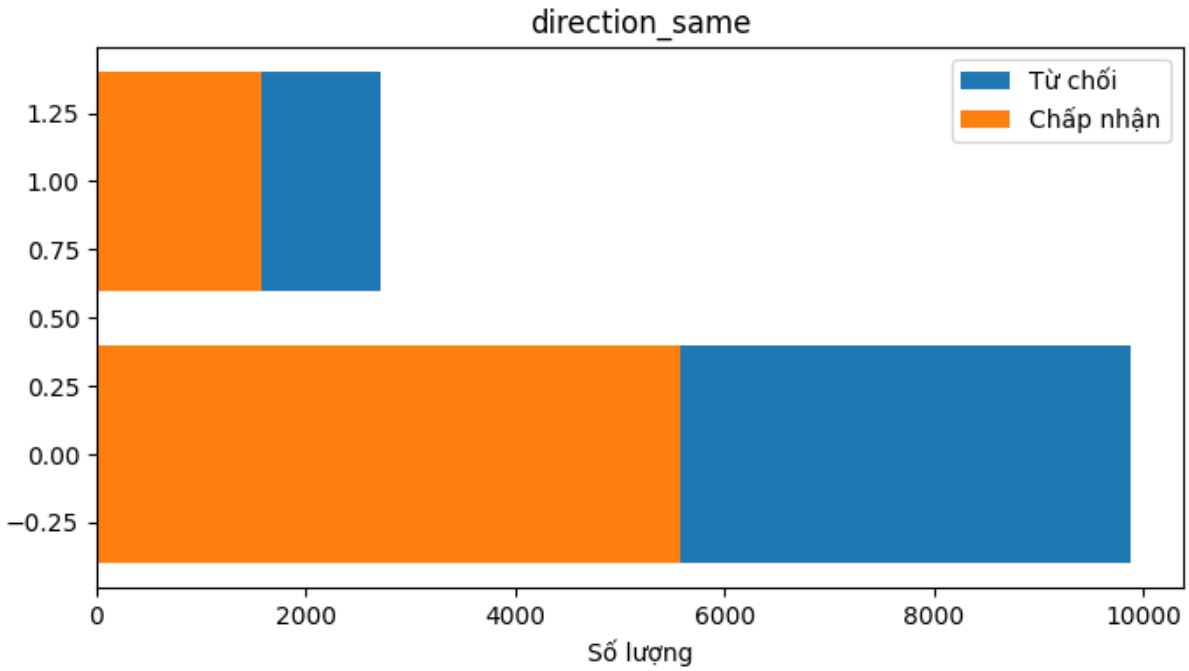
3. InCome:



	income	Tổng số lượng	Chấp nhận	Từ chối	Phần trăm chấp nhận	Phần trăm từ chối
2	$25000 - $37499	2006	1190	816	59.322	40.678
1	$12500 - $24999	1825	1047	778	57.370	42.630
3	$37500 - $49999	1795	1010	785	56.267	43.733
0	$100000 or More	1717	992	725	57.775	42.225
4	$50000 - $62499	1655	984	671	59.456	40.544
8	Less than $12500	1034	612	422	59.188	40.812
7	$87500 - $99999	879	465	414	52.901	47.099
6	$75000 - $87499	856	414	442	48.364	51.636
5	$62500 - $74999	843	443	400	52.550	47.450

- Hầu hết người dùng trong dữ liệu này có thu nhập từ 12500 đô đến 49999 đô. Người dùng có thu nhập cao chấp nhận nhiều phiếu giảm giá hơn những người khác. Ngoài ra người có thu nhập tầm trung hầu hết từ chối phiếu giảm giá.

4. Direction_same



	direction_same	Tổng số lượng	Chấp nhận	Từ chối	Phần trăm chấp nhận	Phần trăm từ chối
0	0	9892	5576	4316	56.369	43.631
1	1	2718	1581	1137	58.168	41.832

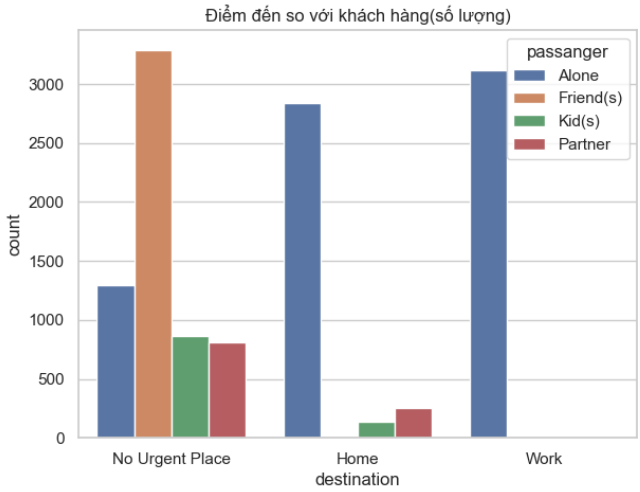
- Đặc tính 'direction_same' có hai giá trị là 0 và 1. Cả hai giá trị này có tỷ lệ chấp nhận và từ chối gần như tương tự nhau. Vì vậy đặc tính này không hữu ích trong dự đoán. Vì vậy, bỏ đặc tính này.

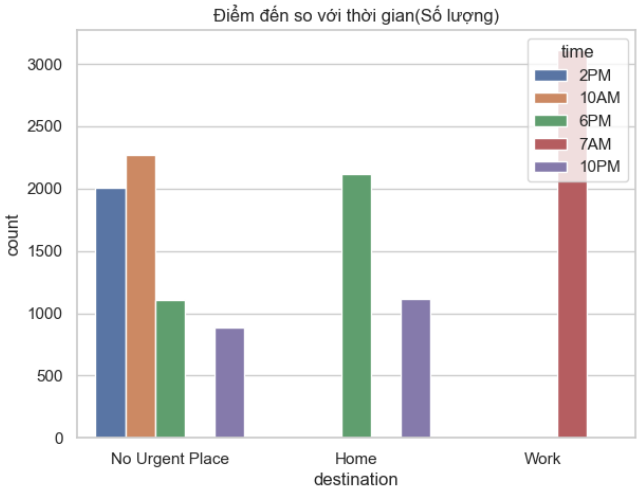
9. Phân tích hai biến số

1. Người dùng đi đến 'Work', 'Home', 'No Urgent Place' lúc nào , với ai ?

```
fig,axes = plt.subplots(1, 2, figsize=(15, 5))
sns.countplot(x = data['destination'],hue=data['passanger'],ax=axes[0])
sns.countplot(x = data['destination'],hue=data['time'],ax=axes[1])
axes[0].set_title('Điểm đến so với khách hàng(số lượng)'); axes[1].set_title
('Điểm đến so với thời gian(Số lượng)')
```

✓ 0.4s





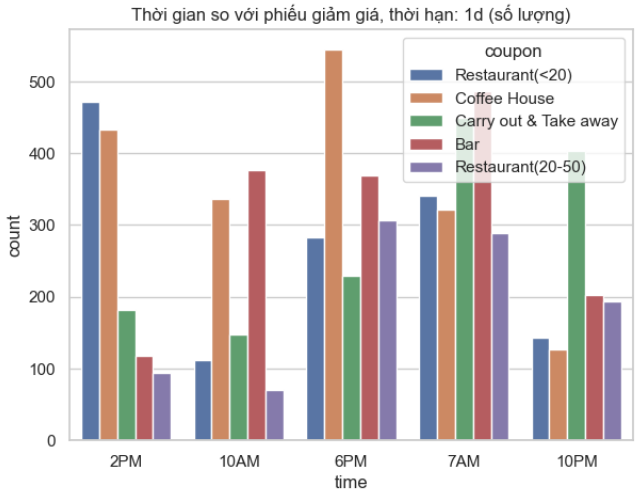
	destination	time	Tổng số lượng	Chấp nhận	Từ chối	Phần trăm chấp nhận	Phần trăm từ chối
2	No Urgent Place	6PM	1102	730	372	66.243	33.757
0	No Urgent Place	2PM	2006	1327	679	66.152	33.848
1	No Urgent Place	10AM	2271	1380	891	60.766	39.234
7	Home	6PM	2118	1151	967	54.344	45.656
13	Work	7AM	3114	1553	1561	49.872	50.128
9	Home	10PM	1112	484	628	43.525	56.475

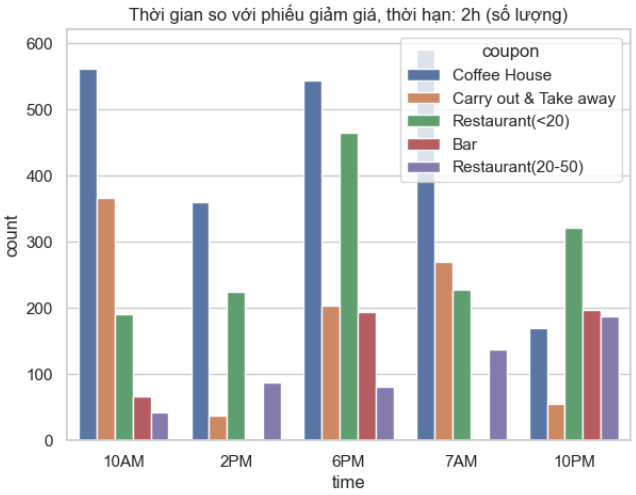
	passanger	time	Tổng số lượng	Chấp nhận	Từ chối	Phần trăm chấp nhận	Phần trăm từ chối
7	Friend(s)	6PM	592	448	144	75.676	24.324
0	Alone	2PM	361	265	96	73.407	26.593
17	Partner	6PM	309	216	93	69.903	30.097
12	Kid(s)	6PM	267	125	142	46.816	53.184
4	Alone	10PM	990	461	529	46.566	53.434
14	Kid(s)	10PM	167	43	124	25.749	74.251

- Người dùng đi cùng bạn bè ‘Friend(s)’ chỉ có điểm đến không khẩn cấp ‘No Urgent Place’.
- Vào lúc 7AM, chỉ người dùng đi làm ‘Work’ mới ra ngoài một mình.
- Vào lúc 2PM, người dùng chỉ đến nơi không khẩn cấp ‘No Urgent Place’
- ‘Home’, người dùng chủ yếu chỉ đi 1 mình vào lúc 6PM và 10PM
- ‘No Urgent Place’, người dùng chủ yếu đi một mình ‘Alone’ lúc 2PM và với bạn ‘Friend(s)’ lúc 6PM
- Người dùng đi cùng bạn bè ‘Friend(s)’ lúc 6PM có tỉ lệ chấp nhận cao nhất
- Người dùng đi một mình ‘Alone’ lúc 2PM cũng có tỉ lệ chấp nhận cao
- Người dùng đi cùng trẻ em lúc 10PM hầu hết từ chối phiếu giảm giá

2. Tỷ lệ chấp nhận và từ chối phiếu giảm giá cao vào thời điểm nào

```
fig,axes = plt.subplots(1, 2, figsize=(15, 5))
sns.countplot(x = data[data.expiration == '1d']['time'],hue=data[data.
expiration == '1d']['coupon'],ax=axes[0])
sns.countplot(x = data[data.expiration == '2h']['time'],hue=data[data.
expiration == '2h']['coupon'],ax=axes[1])
axes[0].set_title('Thời gian so với phiếu giảm giá, thời hạn: 1d (số lượng)');
axes[1].set_title('Thời gian so với phiếu giảm giá, thời hạn: 2h (số lượng)')
```





	coupon	time	Tổng số lượng	Chấp nhận	Từ chối	Phần trăm chấp nhận	Phần trăm từ chối
10	Carry out & Take away	2PM	218	189	29	86.697	13.303
2	Restaurant(<20)	6PM	748	619	129	82.754	17.246
12	Carry out & Take away	6PM	433	357	76	82.448	17.552
0	Restaurant(<20)	2PM	696	567	129	81.466	18.534
18	Bar	7AM	487	178	309	36.550	63.450
16	Bar	10AM	443	160	283	36.117	63.883
15	Bar	2PM	118	40	78	33.898	66.102
24	Restaurant(20-50)	10PM	380	125	255	32.895	67.105

- Vào lúc 7AM và 2PM, người dùng sẽ không nhận phiếu giảm giá bar có thời hạn 2h
- Vào lúc 2PM và 6PM, người dùng chủ yếu chấp nhận phiếu giảm giá 'Restaurant(<20)' và 'Carry out & take away' với tỉ lệ 80%
- Vào lúc 2PM, người dùng chủ yếu chấp nhận phiếu giảm giá 'Carry out & take away' với tỉ lệ 86%
- Từ 7AM đến 6PM. Phiếu giảm giá Coffee Hosue là nhiều nhất
- 10PM chỉ có Carry out & take away(1d) và Restaurant(<20)(2h) là cao nhất.
- Vào 10PM, người dùng không chấp nhận phiếu giảm giá 'Restaurant(20-50)'
- Vào 7AM, 10AM, 2PM người dùng từ chối phiếu giảm giá cho Bar

10. Thiết kế đặc trưng

1. To_Coupon

‘toCoupon_GEQ15min’ và ‘toCoupon_GEQ25min’ cho biết khoảng cách lái xe đến nhà hàng/quán cà phê/quán bar, vì vậy chúng tôi có thể kết hợp chúng và tạo một đặc trưng mới gọi là 'to_coupon'.

Các đặc trưng 'toCoupon_GEQ15min' và 'toCoupon_GEQ25min' có hai giá trị phân loại 0 & 1 và 'to_coupon' có ba giá trị phân loại 0,1 & 2.

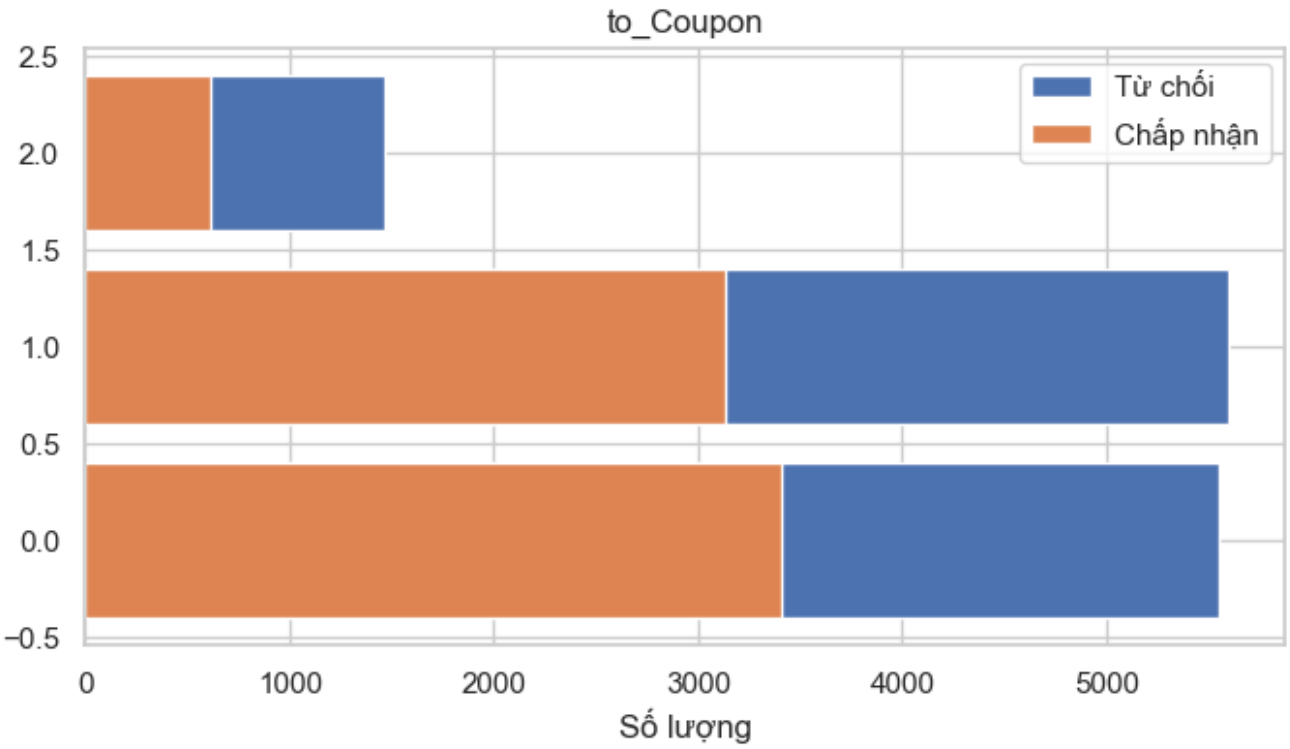
0: khoảng cách lái xe nhỏ hơn hoặc bằng 15 phút

1: khoảng cách lái xe lớn hơn 15 phút và nhỏ hơn hoặc bằng 25 phút

2: khoảng cách lái xe lớn hơn 25 phút

```
to_Coupon = []
for i in tqdm(range(data.shape[0])):
    if (list(data['toCoupon_GEQ15min'])[i] == 0):
        to_Coupon.append(0)
    elif (list(data['toCoupon_GEQ15min'])[i] == 1)and(list(data
    ['toCoupon_GEQ25min'])[i] == 0):
        to_Coupon.append(1)
    else:
        to_Coupon.append(2)

# print(len(to_Coupon)) #12610
data['to_Coupon'] = to_Coupon
print('Unique values:',data['to_Coupon'].unique())
print('-'*50)
data['to_Coupon'].describe()
```

```
100%|██████████| 12610/12610 [00:18<00:00, 673.13it/s]

Unique values: [0 1 2]
--------------------------------------------------



count    12610.000000
mean         0.675813
std          0.671687
min          0.000000
25%          0.000000
50%          1.000000
75%          1.000000
max          2.000000
Name: to_Coupon, dtype: float64
```



	to_Coupon	Tổng số lượng	Chấp nhận	Từ chối	Phần trăm chấp nhận	Phần trăm từ chối
1	1	5596	3136	2460	56.040	43.960
0	0	5551	3407	2144	61.376	38.624
2	2	1463	614	849	41.969	58.031

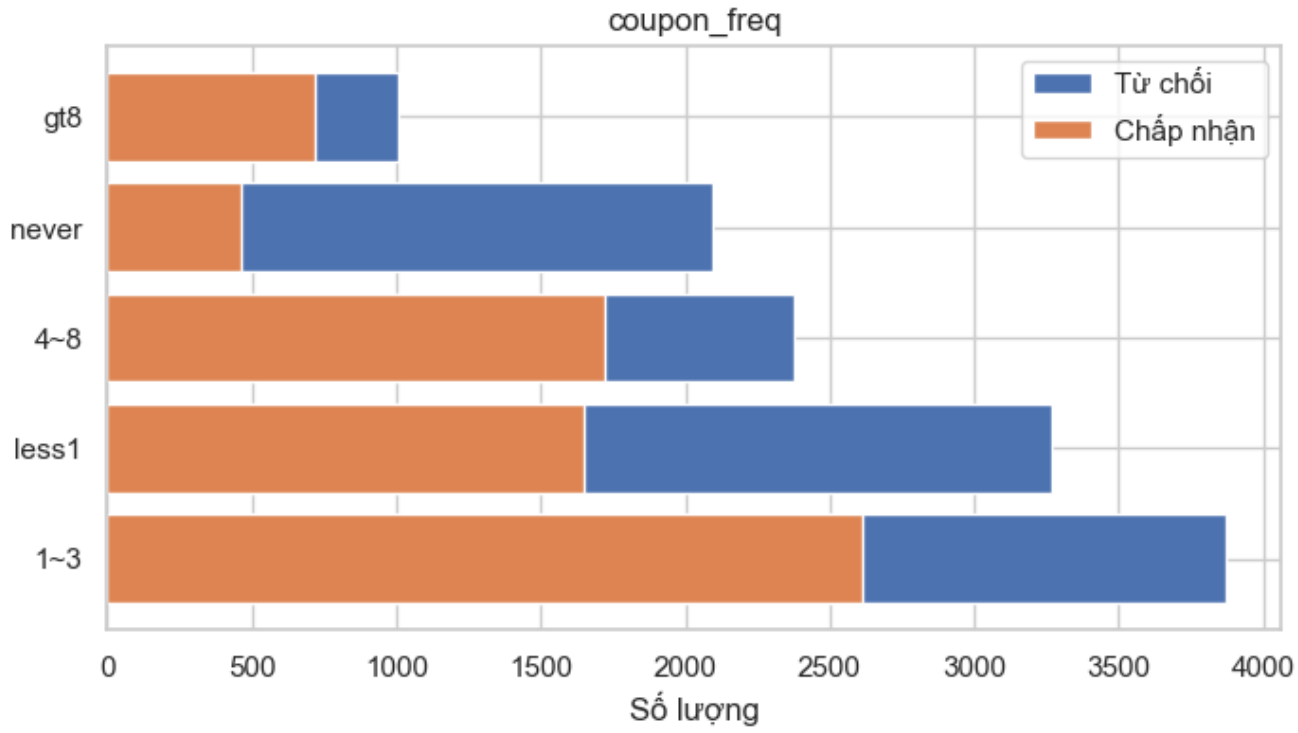
Chúng ta có thể thấy rằng hầu hết các phiếu giảm giá được cung cấp đều ở gần đó trong vòng 25 phút. Ngoài ra, người dùng thường chấp nhận các phiếu giảm giá có khoảng cách dưới 25 phút.

2. Coupon_freq

   'coupon_freq' là sự kết hợp của năm tính năng, RestaurantLessThan20, CoffeeHouse, CarryAway, Bar và Restaurant20To50. Năm tính năng này cho biết tần suất người dùng đến RestaurantLessThan20, CoffeeHouse, CarryAway, Bar và Restaurant20To50.

   Mỗi người dùng được yêu cầu một phiếu giảm giá, vì vậy chúng tôi chỉ cần tần suất của một loại phiếu giảm giá đó, không cần các chi tiết phiếu giảm giá khác. vì vậy chúng tôi tạo một tính năng mới cho biết tần suất người dùng sử dụng phiếu giảm giá đã hỏi.

```
coupon_freq = []
for i in tqdm(range(data.shape[0])):
    if (list(data['coupon'])[i] == 'Restaurant(<20)'):
        coupon_freq.append(list(data['RestaurantLessThan20'])[i])
    elif (list(data['coupon'])[i] == 'Coffee House'):
        coupon_freq.append(list(data['CoffeeHouse'])[i])
    elif (list(data['coupon'])[i] == 'Carry out & Take away'):
        coupon_freq.append(list(data['CarryAway'])[i])
    elif (list(data['coupon'])[i] == 'Bar'):
        coupon_freq.append(list(data['Bar'])[i])
    elif (list(data['coupon'])[i] == 'Restaurant(20-50)'):
        coupon_freq.append(list(data['Restaurant20To50'])[i])

data['coupon_freq'] = coupon_freq
print('Unique values:',data['coupon_freq'].unique())
print('-'*50)
data['coupon_freq'].describe()
```



- Hầu hết người dùng trong dữ liệu này đã truy cập các dịch vụ Nhà hàng(<$20) và Mang đi nhiều lần và họ cũng có tỷ lệ chấp nhận phiếu giảm giá cao nhất.
- Người dùng đã truy cập Coffee House 1 đến 3 lần và 4 đến 8 lần cũng có tỷ lệ chấp nhận phiếu giảm giá cao hơn.
- Hầu hết người dùng chỉ ghé thăm Nhà hàng đắt tiền một lần và những người dùng đã ghé thăm Nhà hàng đắt tiền hơn 4 lần có tỷ lệ chấp nhận phiếu giảm giá cao nhất cho Nhà hàng đắt tiền.
- Hầu hết người dùng trong dữ liệu này chưa bao giờ ghé thăm một Quán bar và hầu hết trong số họ từ chối phiếu giảm giá của quán bar và những người dùng đã ghé thăm Quán bar 1 đến 3 lần hoặc 4 đến 8 lần có tỷ lệ chấp nhận phiếu giảm giá cao hơn.

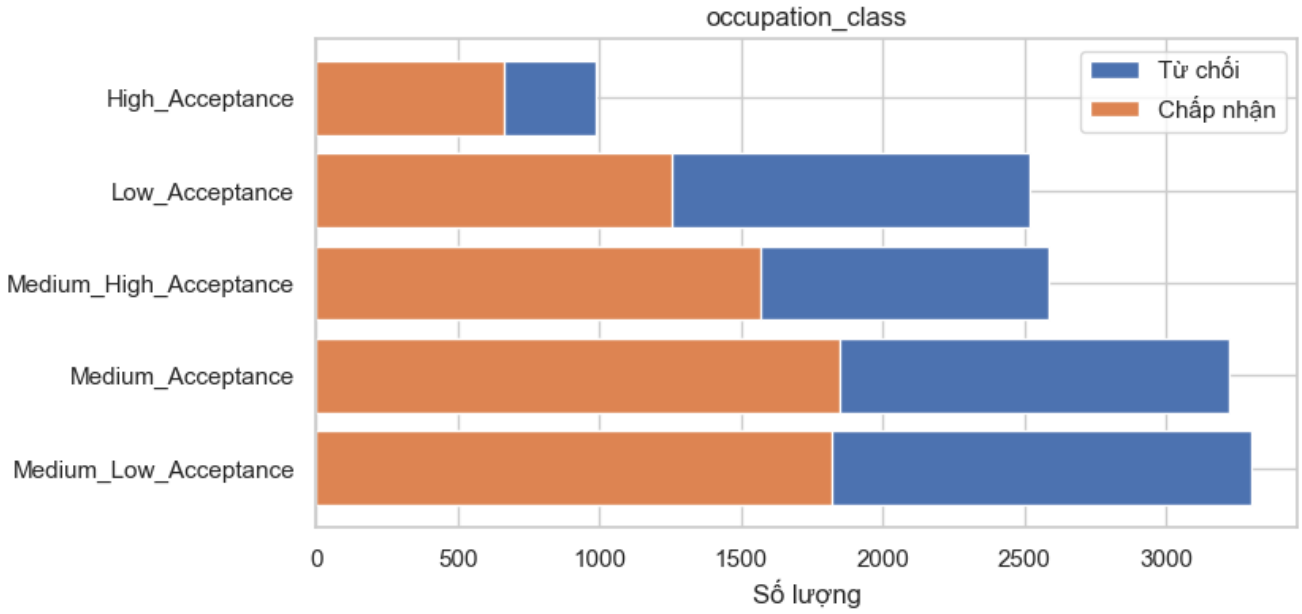
Điều đó có nghĩa là coupon_freq có liên quan nhiều đến biến mục tiêu.

3. occupation_class

Đặc điểm occupation_class có 25 không có giá trị riêng biệt, điều này tạo ra rất ít trong ma trận dữ liệu sau khi mã hóa nên chúng tôi có thể cố gắng chuyển đổi nó thành một đặc điểm mới bằng cách sử dụng kỹ thuật đặc trưng.

Chúng tôi sẽ thử các biến thể khác nhau của việc phân loại các biến số nghề nghiệp, một là chia 25 nghề nghiệp thành năm loại từ kiến thức lĩnh vực hoặc nghề nghiệp của họ nhưng cách đó không hiệu quả lắm. vì vậy chúng tôi sẽ chia chúng thành năm loại cao, medium_high, medium, medium_low và low theo thứ tự tỷ lệ chấp nhận phiếu giảm giá và tạo một tính năng mới có tên là class_class.

```
occupation_dict = {'Healthcare Support':'High_Acceptance','Construction & Extraction':'High_Acceptance','Healthcare Practitioners &
Technical':'High_Acceptance',
                   'Protective Service':'High_Acceptance','Architecture & Engineering':'High_Acceptance','Production Occupations':'Medium_High_Acceptance
                    'Student':'Medium_High_Acceptance','Office & Administrative Support':'Medium_High_Acceptance','Transportation & Material
                    Moving':'Medium_High_Acceptance',
                    'Building & Grounds Cleaning & Maintenance':'Medium_High_Acceptance','Management':'Medium_Acceptance','Food Preparation & Serving
                    Related':'Medium_Acceptance',
                   'Life Physical Social Science':'Medium_Acceptance','Business & Financial':'Medium_Acceptance','Computer &
                   Mathematical':'Medium_Acceptance',
                    'Sales & Related':'Medium_Low_Acceptance','Personal Care & Service':'Medium_Low_Acceptance','Unemployed':'Medium_Low_Acceptance',
                   'Farming Fishing & Forestry':'Medium_Low_Acceptance','Installation Maintenance & Repair':'Medium_Low_Acceptance','Education&Training&
                   Library':'Low_Acceptance',
                    'Arts Design Entertainment Sports & Media':'Low_Acceptance','Community & Social Services':'Low_Acceptance','Legal':'Low_Acceptance',
                    'Retired':'Low_Acceptance'}
# occupation_dict
data['occupation_class'] = data['occupation'].map(occupation_dict)
print('Unique values:',data['occupation_class'].unique())
print('-'*50)
data['occupation_class'].describe()
```



11. Mã hóa dữ liệu

Hầu hết các thuật toán học máy không thể hoạt động trực tiếp trên dữ liệu phân loại. Chúng yêu cầu tất cả các biến đầu vào và biến đầu ra phải là số. Hầu hết các hiệu suất của thuật toán khác nhau dựa trên phương pháp mã hóa nào được sử dụng. Có nhiều cách để mã hóa dữ liệu phân loại.

1. Mã hóa tần số (Frequency Encoding)

   Giá trị thay thế nó là tần suất giá trị đó xuất hiện trong cột

   Trong mã hóa này, chúng tôi sẽ chuyển đổi các đặc trưng phân loại thành giá trị tần số của các danh mục của chúng.

   Dưới đây là các bước để tính giá trị tần số của tính năng:

   1. Nhóm theo biến phân loại và thu được số lượng của từng loại
   2. Để chuẩn hóa chúng chia với tổng số của một biến phân loại
   3. Bản đồ giá trị tần số với các tính năng đào tạo

```
X_train_frequency_encoding = pd.DataFrame()
for i in range(X_train.shape[1]):
  X_train_frequency_encoding[X_train.columns.values[i]+'_freq_enc'] =
  frequency_enc(X_train.columns.values[i],X_train)

print('X_train_frequency_encoding:',X_train_frequency_encoding.shape)
X_train_frequency_encoding.head(3)
```

✓ 0.7s

	destination_freq_enc	passanger_freq_enc	weather_freq_enc	temperature_freq_enc	tin
1051	0.245738	0.572958	0.791634	0.30343	
5799	0.255452	0.572958	0.791634	0.51477	
291	0.498810	0.080987	0.791634	0.30343	

```
X_train_frequency_encoding = pd.DataFrame()
for i in range(X_train.shape[1]):
  X_train_frequency_encoding[X_train.columns.values[i]+'_freq_enc'] =
  frequency_enc(X_train.columns.values[i],X_train)

print('X_train_frequency_encoding:',X_train_frequency_encoding.shape)
X_train_frequency_encoding.head(3)
```

✓ 0.6s

```
X_train_frequency_encoding: (10088, 16)
```

```
X_test_frequency_encoding = pd.DataFrame()
for i in range(X_test.shape[1]):
  X_test_frequency_encoding[X_test.columns.values[i]+'_freq_enc'] =
  frequency_enc(X_test.columns.values[i],X_test)

print('X_test_frequency_encoding:',X_test_frequency_encoding.shape)
X_test_frequency_encoding.head(3)
```

```
X_test_frequency_encoding: (2522, 16)
```

	destination_freq_enc	passanger_freq_enc	weather_freq_enc	temperature_fr
10641	0.499207	0.575734	0.795004	0.
4664	0.499207	0.085646	0.795004	0.
531	0.499207	0.073751	0.795004	0.

12. Models

Sau khi thực hiện phân tích dữ liệu thăm dò và làm sạch dữ liệu, chúng tôi cũng đã thực hiện một số kỹ thuật đặc trưng và thêm đặc trưng mới 'to_coupon', 'coupon_freq' và 'occupation_class' từ các đặc trưng hiện có. Chúng tôi cũng đã chuyển đổi các tính năng phân loại thành các tính năng số bằng cách sử dụng năm phương pháp mã hóa khác nhau và chuẩn bị năm ma trận dữ liệu kiểm tra và huấn luyện khác nhau. Hiện chúng tôi đã sẵn sàng áp dụng thuật toán học máy cho ma trận dữ liệu đã chuẩn bị của mình.

Có nhiều thuật toán học máy để thử nghiệm khi bạn đang giải quyết vấn đề phân loại. Bạn không bao giờ biết trước mô hình phân loại nào sẽ phù hợp nhất với mình. Vì vậy, chúng tôi sẽ thử nghiệm với một vài mô hình phân loại và kiểm tra xem mô hình nào hoạt động tốt nhất. Việc thực hiện điều chỉnh siêu tham số để tối đa hóa hiệu suất của mô hình cũng rất quan trọng.

1. Logistic Regression

```
def Logistic_Regression(x_train,y_train,x_test,y_test):


    clf = LogisticRegression(random_state=0,C=1.0)
    parameters = {'C':[0.01, 0.1, 1, 10, 100, 500]}
    model = RandomizedSearchCV(clf, parameters, cv=5, scoring='roc_auc')
    model.fit(x_train, y_train)
    best_C = model.best_params_['C']

    clf = LogisticRegression(random_state=0,C=best_C).fit(x_train, y_train)

    Train_loss = log_loss(y_train,clf.predict_proba(x_train))
    Train_AUC = roc_auc_score(y_train,clf.predict_proba(x_train)[:,1])
    Test_loss = log_loss(y_test,clf.predict_proba(x_test))
    Test_AUC = roc_auc_score(y_test,clf.predict_proba(x_test)[:,1])

    return best_C,Train_loss,Train_AUC,Test_loss,Test_AUC
```

```
best_C_FreEnc, Train_loss_FreEnc, Train_AUC_FreEnc, Test_loss_FreEnc, Test_AUC_FreEnc = Logistic_Regression
(X_train_frequency_encoding,y_train,X_test_frequency_encoding,y_test)
```

```
summary_table = PrettyTable(["Model","Encoding", "Hyperparameter1", "Hyperparameter2", "Train_log_loss", "Train_roc_auc_score", "Test_log_loss",
"Test_roc_auc_score"]) #heading

summary_table.add_row(["Logistic Regression","Frequency Encoding",best_C_FreEnc,'',round(Train_loss_FreEnc,3),round(Train_AUC_FreEnc,3),round
(Test_loss_FreEnc,3),round(Test_AUC_FreEnc,3)])

table = pd.read_html(summary_table.get_html_string())
Logistic_Regression_Result = table[0]
Logistic_Regression_Result
```

	Model	Encoding	Hyperparameter1	Hyperparameter2	Train_log_loss	Train_roc_auc_score	Test_log_loss	Test_roc_auc_score
0	Logistic Regression	Frequency Encoding	10	NaN	0.651	0.644	0.655	0.633

2. K-Nearest Neighbor

```
def K_Neighbors_Classifier(x_train,y_train,x_test,y_test):


  clf = KNeighborsClassifier()
  parameters = {'n_neighbors':[11, 15, 21, 31, 41, 51]}
  model = RandomizedSearchCV(clf, parameters, cv=5, scoring='roc_auc')
  model.fit(x_train, y_train)
  best_n_neighbors = model.best_params_['n_neighbors']

  clf = KNeighborsClassifier(n_neighbors=best_n_neighbors).fit(x_train, y_train)

  Train_loss = log_loss(y_train,clf.predict_proba(x_train))
  Train_AUC = roc_auc_score(y_train,clf.predict_proba(x_train)[:,1])
  Test_loss = log_loss(y_test,clf.predict_proba(x_test))
  Test_AUC = roc_auc_score(y_test,clf.predict_proba(x_test)[:,1])

  return best_n_neighbors,Train_loss,Train_AUC,Test_loss,Test_AUC
```

```
best_n_FreEnc, Train_loss_FreEnc, Train_AUC_FreEnc, Test_loss_FreEnc, Test_AUC_FreEnc = K_Neighbors_Classifier(X_train_frequency_encoding,y_train,
X_test_frequency_encoding,y_test)
```

```
summary_table = PrettyTable(["Model","Encoding", "Hyperparameter1", "Hyperparameter2", "Train_log_loss", "Train_roc_auc_score", "Test_log_loss",
"Test_roc_auc_score"])

summary_table.add_row(["K-Nearest Neighbor","Frequency Encoding",best_n_FreEnc,'',round(Train_loss_FreEnc,3),round(Train_AUC_FreEnc,3),round
(Test_loss_FreEnc,3),round(Test_AUC_FreEnc,3)])

table = pd.read_html(summary_table.get_html_string())
K_Nearest_Neighbor_Result = table[0]
K_Nearest_Neighbor_Result
```

	Model	Encoding	Hyperparameter1	Hyperparameter2	Train_log_loss	Train_roc_auc_score	Test_log_loss	Test_roc_auc_score
0	K-Nearest Neighbor	Frequency Encoding	15	NaN	0.57	0.76	0.735	0.671

3. Decision Tree

```
def Decision_Tree_Classifier(x_train,y_train,x_test,y_test):


  clf = DecisionTreeClassifier(class_weight='balanced')
  parameters = {'max_depth':[1, 5, 10, 50], 'min_samples_split':[5, 10, 100, 500]}
  model = RandomizedSearchCV(clf, parameters, cv=5, scoring='roc_auc')
  model.fit(x_train, y_train)
  best_depth = model.best_params_['max_depth']
  best_samples_split = model.best_params_['min_samples_split']

  clf = DecisionTreeClassifier(class_weight='balanced', max_depth=best_depth, min_samples_split=best_samples_split, random_state=0)
  clf.fit(x_train, y_train)

  Train_loss = log_loss(y_train,clf.predict_proba(x_train))
  Train_AUC = roc_auc_score(y_train,clf.predict_proba(x_train)[:,1])
  Test_loss = log_loss(y_test,clf.predict_proba(x_test))
  Test_AUC = roc_auc_score(y_test,clf.predict_proba(x_test)[:,1])

  return best_depth,best_samples_split,Train_loss,Train_AUC,Test_loss,Test_AUC
```

```
best_depth_FreEnc,best_samples_split_FreEnc,Train_loss_FreEnc,Train_AUC_FreEnc,Test_loss_FreEnc,Test_AUC_FreEnc = Decision_Tree_Classifier
(X_train_frequency_encoding,y_train,X_test_frequency_encoding,y_test)
```

```
summary_table = PrettyTable(["Model","Encoding", "Hyperparameter1", "Hyperparameter2", "Train_log_loss", "Train_roc_auc_score", "Test_log_loss",
"Test_roc_auc_score"])

summary_table.add_row(["Decision Tree","Frequency Encoding",best_depth_FreEnc,best_samples_split_FreEnc,round(Train_loss_FreEnc,3),round(Train_AUC_FreEnc,
3),round(Test_loss_FreEnc,3),round(Test_AUC_FreEnc,3)])

table = pd.read_html(summary_table.get_html_string())
Decision_Tree_Result = table[0]
Decision_Tree_Result
```

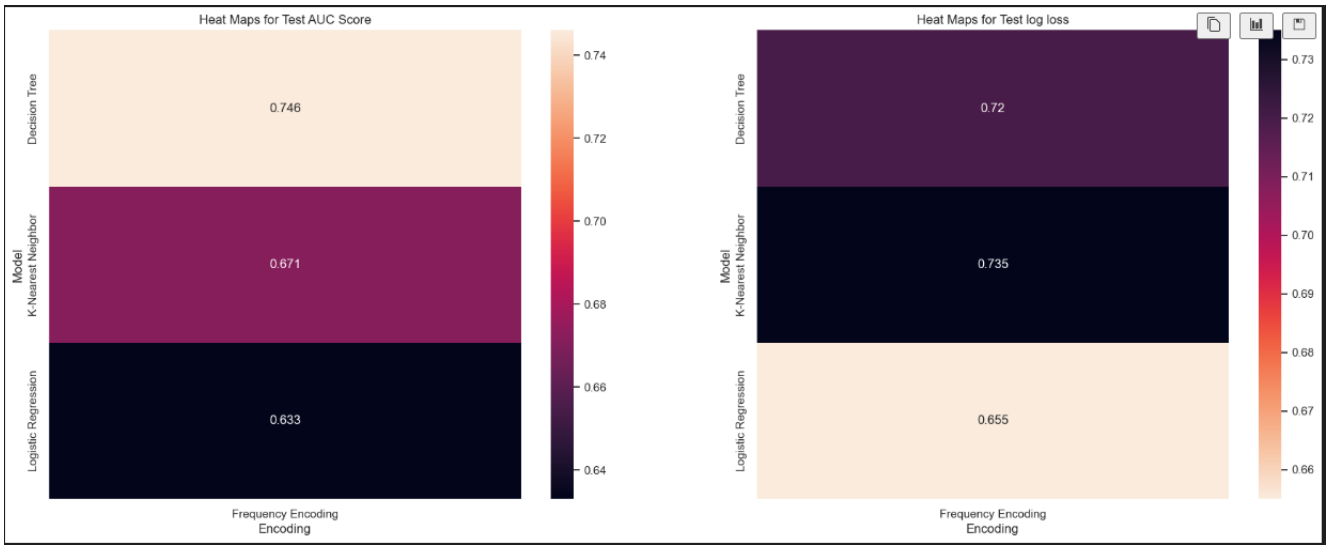
	Model	Encoding	Hyperparameter1	Hyperparameter2	Train_log_loss	Train_roc_auc_score	Test_log_loss	Test_roc_auc_score
0	Decision Tree	Frequency Encoding	10	100	0.533	0.806	0.72	0.746

4. So Sánh

```
Model_Result = [Logistic_Regression_Result,K_Nearest_Neighbor_Result,Decision_Tree_Result]
Result = pd.concat(Model_Result,ignore_index=True)
(Result).sort_values(by=['Test_roc_auc_score'],ascending=False).head(10)
```

	Model	Encoding	Hyperparameter1	Hyperparameter2	Train_log_loss	Train_roc_auc_score	Test_log_loss	Test_roc_auc_score
2	Decision Tree	Frequency Encoding	10	100.0	0.533	0.806	0.720	0.746
1	K-Nearest Neighbor	Frequency Encoding	15	NaN	0.570	0.760	0.735	0.671
0	Logistic Regression	Frequency Encoding	10	NaN	0.651	0.644	0.655	0.633

```
fig, ax = plt.subplots(1,2, figsize=(22,8))
sns.heatmap(Result.pivot('Model','Encoding','Test_roc_auc_score'), annot = True, fmt='.3g', ax=ax[0])
sns.heatmap(Result.pivot('Model','Encoding','Test_log_loss'), annot = True, fmt='.3g', cmap=sns.cm.rocket_r, ax=ax[1])
ax[0].set_title('Heat Maps for Test AUC Score')
ax[1].set_title('Heat Maps for Test log loss')
plt.show()
```



- Chúng ta có thể so sánh test log loss và test AUC score cảu từng mô hình này để tìm ra mô hình tốt nhất.
- Chúng ta có thể thấy Decision Tree hoạt động tốt nhất so với các mô hình khác.

CHƯƠNG III: Tài liệu tham khảo

1. (Nguồn tham khảo: Từ Minh Phương, Giáo trình Nhập môn trí tuệ nhân tạo, 2014)