ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KINH TẾ LUẬT

KHOA HỆ THỐNG THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KỲ
PHÂN TÍCH DỮ LIỆU VỚI R/PYTHON

ĐỀ TÀI: XÂY DỰNG MÔ HÌNH DỰ BÁO GIÁ VÉ MÁY BAY

Giảng viên hướng dẫn: ThS. Nguyễn Phát Đạt

Nhóm: Bó Đũa

Thành phố Hồ Chí Minh, 28 tháng 11 năm 2022

DANH SÁCH THÀNH VIÊN

STT	Họ và tên	MSSV
1	Võ Nguyên Trúc Lâm	K204110572
2	Nguyễn Hữu Thuận	K194060821
3	Lê Thảo Giang	K204110562
4	Lê Phước Toàn	K204110586
5	Nguyễn Thị Bảo Trâm	K204110588
6	Thẩm Thị Tú Uyên	K204111792

LỜI CẢM ƠN

Nhóm Bó Đũa xin gửi lời cảm ơn chân thành đến t**hầy Nguyễn Phát Đạt** - giảng viên giảng dạy môn *Phân tích dữ liệu với R/Python* vì đã tận tình hướng dẫn, truyền đạt những kiến thức quý báu cho tụi em trong suốt thời gian học tập vừa qua cũng như giải đáp những thắc mắc của nhóm trong thời gian thực hiện đồ án. Những kiến thức tích lũy trong quá trình học tập, chính là cơ sở để chúng em hoàn thành đồ án này.

Trong quá trình thực hiện đồ án, mặc dù nhóm đã cố gắng hết sức nhưng chắc chắn bài làm khó có thể tránh khỏi những thiếu sót, nhóm rất mong thầy xem xét và góp ý để bài làm của chúng em được tốt hơn và có thể ứng dụng được trong tương lai.

Một lần nữa, nhóm Bó Đũa xin chân thành cảm ơn thầy.

Nhóm Bó Đũa

LỜI CAM KẾT

Nhóm xin cam kết đồ án do chính nhóm thực hiện dưới sự hướng dẫn thầy Nguyễn Phát Đạt. Trong quá trình thực hiện đề tài, nhóm xin cam kết thực hiện đúng các quy định, các số liệu và kết quả trình bày trong báo cáo là chính xác. Tất cả các tài liệu tham khảo trên Internet, sách và giáo trình đều được trích dẫn cụ thể.

Thành phố Hồ Chí Minh, 28 tháng 11 năm 2022

Nhóm Bó Đũa

MỤC LỤC

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH ẢNH

DANH MỤC TỪ VIẾT TẮT

DB	Digital Business
KNN	K-Nearest Neighbors
RF	Random Forest
R^2	R-squared: Hệ số xác định R bình phương
MSE	Mean Squared Error: Sai số bình phương trung bình
RMSE	Root Mean Squared Error: Sai số bình phương trung bình gốc
MAE	Mean Absolute Error: Giá trị sai số tuyệt đối trung bình
RMSLE	Root Mean Squared Logarithmic Error: Sai số bình phương trung bình gốc của các giá trị dự đoán và thực tế được chuyển đổi logarit
MAPE	Mean Absolute Percentage Error: Trung bình độ lệch tương đối

TỔNG QUAN ĐỒ ÁN

1. Lý do chọn đề tài:

Ngày nay, ngày càng nhiều người lựa chọn máy bay làm phương tiện di chuyển bởi sự tiện lợi và tốc độ nhanh chóng so với các phương tiện khác. Với sự phát triển mạnh mẽ của công nghệ thông tin, các hãng hàng không đều cung cấp vé thông qua rất nhiều kênh khác nhau như trang web chính thức của hãng hàng không, các đại lý vé máy bay, các công ty du lịch. Người mua vé máy bay có thể dễ dàng tìm kiếm thông tin vé của bất kỳ hãng hàng không nào trên thế giới một cách dễ dàng mà không cần phải rời khỏi nhà.

Ngành hàng không dân dụng ở Ấn Độ đã nổi lên như một trong những ngành phát triển nhanh nhất ở nước này trong ba năm qua và có thể được phân loại thành dịch vụ vận tải hàng không theo lịch trình bao gồm các hãng hàng không trong nước và quốc tế, dịch vụ vận tải hàng không không theo lịch trình bao gồm thuê chuyến. Giao thông nội địa đóng góp khoảng 69% tổng lưu lượng hàng không ở Nam Á và công suất sân bay của Ấn Độ dự kiến sẽ xử lý 1 tỷ chuyến đi hàng năm vào năm 2023. Ngành hàng không Ấn Độ đã phục hồi hoàn toàn sau cú sốc đại dịch covid-19 được thể hiện qua lưu lượng hàng không đạt 613.566 trong quý đầu tiên của năm tài chính 2022-23 so với 300.405 trong cùng kỳ năm ngoái, tăng 104,24%. Ấn Độ hiện là thị trường hàng không dân dụng lớn thứ 7 thế giới và dự kiến sẽ trở thành thị trường hàng không dân dụng lớn thứ 3 trong vòng 10 năm tới.

Thị trường hàng không nội địa là một thị trường cạnh tranh khốc liệt với mức tăng trưởng hàng năm cao. Các hãng hàng không thường được biết đến với việc sử dụng các thuật tốn thương mại vô cùng phức tạp để tính toán giá vé máy bay sao cho tối đa hóa lợi nhuận thu được, trong khi đó hành khách tìm kiếm giá vé máy bay rẻ nhất nhằm tiết kiệm cho túi tiền của mình. Trong mối quan hệ đối nghịch giữa các hãng

hàng không và hành khách đó thì lợi thế lại nghiêng về các hãng hàng không, các hãng hàng không có trong tay một lượng rất lớn dữ liệu về giá vé máy bay và nhu cầu đi lại trong quá khứ để từ đó có thể dự báo được nhu cầu đi lại của hành khách trong tương lai và đưa ra giá vé có lợi nhất về mặt lợi nhuận, trong khi đó hành khách thường chỉ có một lựa chọn phổ biến đó là mua vé càng sớm càng tốt để có được giá vé tốt vì thường giá vé sẽ tăng cao khi ngày mua cận kề với ngày khởi hành, tuy nhiên thực tế đã chứng minh rằng điều này không hoàn toàn đúng khi các hãng hàng không có thể linh hoạt điều chỉnh giá vé tùy theo tình hình thực tế của từng chuyến bay cụ thể. Trong ngành công nghiệp du lịch, các công ty lữ hành và các hãng hàng không sử dụng chiến lược định giá phân biệt một cách thường xuyên để áp đặt các mức giá khác nhau cho các đối tượng khách hàng khác nhau. Với các hãng hàng không, với cùng một loại chỗ ngồi trên một chuyến bay, nhưng giá dành cho đối tượng khách hàng là doanh nhân (bận rộn và ít có sự linh hoạt về thời gian nhưng lại có khả năng chi trả cao và ít quan tâm về giá) sẽ cao hơn đối tượng khách hàng bình thường không có q nhiều ràng buộc về thời gian nhưng lại rất nhạy cảm về giá vé. Ngoài ra các hãng hàng không cũng có thể áp dụng giảm giá cho các vé được mua sớm hoặc cận ngày tùy theo tình hình thực tế của từng chuyến bay cụ thể.

Do đặc thù của ngành hàng không, hành khách thường phải bỏ ra số tiền khác nhau chỉ để mua các vé máy bay cho cùng chặng bay, cùng loại chỗ ngồi, cùng chất lượng dịch vụ. Hành khách thường dựa theo tâm lý chung trong việc mua vé máy bay như: mua càng sớm thì giá vé càng rẻ hoặc càng gần ngày mua thì giá vé càng mắc. Tuy nhiên đây chỉ là tâm lý chung thường thấy mà không dựa trên bất kỳ một cơ sở khoa học nào và trong thực tế đã chứng minh ngược lại. Đề tài này sẽ làm rõ vấn đề này và đưa ra được một cơ sở vững vàng hơn trong việc hỗ trợ hành khách chọn mua vé phù hợp. Đồng thời đề tài này cũng tìm hiểu trong số các thuộc tính thu thập được trong bộ dữ liệu, thuộc tính nào có ảnh hưởng lớn nhất tới sự biến động của giá vé máy bay.

2. Mục tiêu thực hiện đồ án:

Mục tiêu của nghiên cứu là kiểm tra tập dữ liệu đặt vé chuyến bay và chạy các thử nghiệm giả thuyết thống kê khác nhau để trích xuất thông tin hữu ích. Tập dữ liệu sẽ được huấn luyện bằng cách sử dụng thuật toán "Random Forest, Decision Tree, K-Nearest Neighbors" để dự báo một biến mục tiêu liên tục. Để đánh giá những kết quả đạt được, đánh giá mức độ chính xác của việc dự đoán, nhóm đã sử dụng các chỉ số: R-squared, MSE, RMSE, MAE... Hành khách sẽ được hưởng lợi rất nhiều từ những hiểu biết độc đáo được phát hiện thông qua những phân tích nghiên cứu trên tập dữ liệu này. Mục tiêu của đề tài là nghiên cứu dự báo vé máy bay trong một khoảng thời gian nào đó, ứng dụng cho người dùng khi có nhu cầu đặt vé máy bay cho hành trình của họ.

3. Câu hỏi nghiên cứu:

Nhóm của chúng em sẽ tìm kiếm và giải quyết bài toán bằng cách đặt các câu hỏi sau:

- Các mô hình Random Forest, K-Nearest Neighbors, Decision Tree thực hiện dự báo giá chuyến bay như thế nào?
- Giá vé máy bay giữa hạng Thương gia và hạng Phổ thông khác nhau như thế nào?
- Chi phí thay đổi như thế nào nếu vé được mua chỉ một hoặc hai ngày trước ngày khởi hành?
- Giá vé có thay đổi theo thời gian khởi hành và thời gian đến không?
- Khi xuất hiện các trạm dừng thì giá vé sẽ ảnh hưởng như thế nào?
- Chi phí thay đổi như thế nào khi điểm đi và điểm đến bị thay đổi?

4. Đối tượng và phạm vi nghiên cứu đồ án:

4.1. Đối tượng nghiên cứu

Đối tượng nghiên cứu: giá vé máy bay thuộc các hãng hàng không tại Ấn Độ trong thời gian 50 ngày tại công ty du lịch (booking website) chuyên cung cấp vé máy bay - 'Ease my trip'. Nhóm dự báo giá vé máy bay sẽ dựa trên các biến về khoảng thời gian, số ngày còn lại, thời điểm, số trạm dừng… của các chuyến bay thuộc các hãng. Và để làm được điều đó, nhóm đã nghiên cứu và vận dụng các thuật toán hồi quy thuộc học máy có giám sát và các chỉ số để đánh giá lại các mô hình thuật toán.

4.2. Phạm vi nghiên cứu

4.2.1. Phạm vi không gian:

Phạm vi nghiên cứu về không gian là thị trường vé máy bay tại Ấn Độ, một thị trường khá năng động và có sự biến đổi lớn theo thời gian. Công ty du lịch "Ease my trip" - là một trong những công ty du lịch hàng đầu của Ấn Độ và là một cái tên đáng tin cậy trong ngành du lịch Ấn Độ. Đối tượng hướng đến cho việc ứng dụng nghiên cứu là khách hàng thuộc hệ phổ thông - thương gia và có nhu cầu mua vé máy bay nội địa tại Ấn Độ.

4.2.2. Phạm vi thời gian:

Mốc thời gian của tập dữ liệu là gần 50 ngày từ ngày 11 tháng 2 đến 31 tháng 3 năm 2022.

5. Phương pháp nghiên cứu:

Nghiên cứu được thực hiện bằng phương pháp nghiên cứu định lượng. Dữ liệu được lựa chọn trên trang web Kaggle. Sau khi thu thập dữ liệu, dữ liệu được xử lý bằng ngôn ngữ Python sử dụng các thư viện được xây dựng cho phân tích dữ liệu, Sau đó các nhân tố được rút trích từ tập dữ liệu sẽ được đưa vào phân tích hồi quy nhằm đánh giá mô hình đề xuất và kiểm định các giả thuyết, sau đó đưa ra kết luận về kết quả thực nghiệm.

6. Kết quả dự kiến:

Từ tập dữ liệu đã thu thập được, mục đích của bài nghiên cứu là dự báo giá vé máy bay dựa vào các yếu tố liên quan và sử dụng thuật toán học máy có giám sát, các chỉ số để đánh giá kết quả. Dựa vào thời gian đi - đến, địa điểm, số trạm dừng, hạng vé và biến giá cả cùng với sự áp dụng của thuật toán để dự báo vé máy bay tại một công ty du lịch ở đây Ease my trip. Việc này sẽ hỗ trợ người dùng (người cần mua vé máy bay) xác định được giá vé trong tương lai một cách ổn định và chính xác trước khi mua. Bài toán sử dụng các thuật toán Random Forest, Decision Tree, KNN để đưa ra kết quả và có sự đánh giá, so sánh kết quả giữa chúng từ đó giúp kết quả đầu ra có cơ sở đánh giá. Giá vé máy bay thường sẽ có sự thay đổi khá nhiều phụ thuộc vào nhiều yếu tố, tuy nhiên yếu tố thời gian so với giờ bay thực tế sẽ góp phần quyết định khá lớn vì vậy khách hàng cần có cơ sở để tham khảo giá vé từ đó điều chỉnh thời gian mua, loại vé, hãng hàng không phù hợp. Kết quả đầu ra là giá vé thực tế của chuyến bay dựa trên các yếu tố liên quan và chỉ số đánh giá độ chính xác của mô hình là trên 90%.

7. Công cụ và ngôn ngữ lập trình sử dụng:

- Công cụ: Visual studio và Google colab
- Ngôn ngữ lập trình: Python

8. Cấu trúc đồ án:

Đồ án bao gồm 4 chương và phần tổng quan đề tài. Phần đầu tiên là phần tổng quan toàn bộ đồ án với các thông tin về lý do, đối tượng, phạm vi nghiên cứu,... liên quan đến vấn đề chính của đồ án. Nội dung chương 1 là cơ sở lý thuyết, tại đây nhóm đã trình bày ý nghĩa, cách thức hoạt động của các thuật ứng dụng vào bài; cùng với đó là các chỉ số để đánh giá mô hình. Thu thập và tiền xử lý dữ liệu là nội dung của chương 2. Nội dung chính của chương tiếp theo là thực nghiệm và đánh giá mô hình. Ở chương này, nhóm sẽ ứng dụng các thuật toán vào bộ dữ liệu đã qua xử lý sau đó dùng các chỉ số đánh giá để quan sát hiệu quả của mô hình. Và phần cuối cùng là thảo luận, phương hướng phát

triển cho những nghiên cứu tiếp theo, tại đây là phần nhận xét, kết luận rút ra từ insights ở chương 3 và đưa ra một số ưu, nhược điểm của mô hình. Từ đó, đề xuất một số hướng phát triển trong tương lai.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

Tóm tắt chương 1:

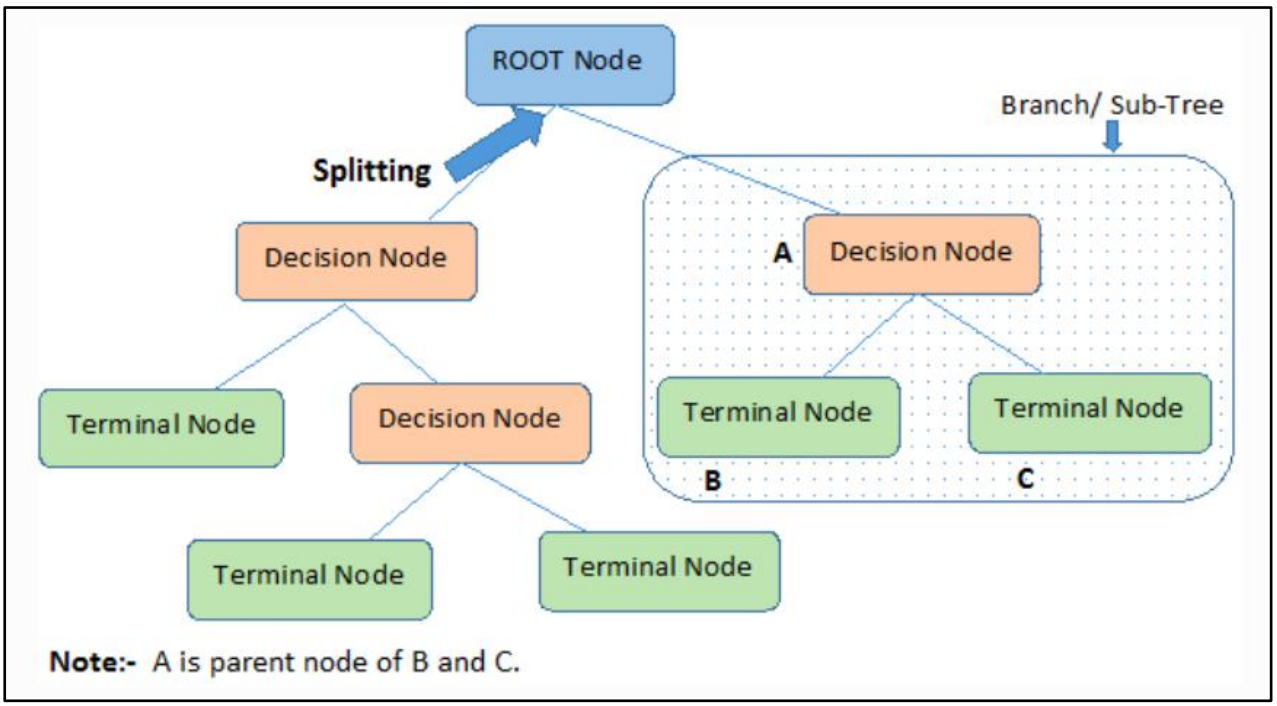
Trình bày tổng quan về khái niệm, ý nghĩa, cách thức hoạt động, lợi ích và công thức tính toán của các thuật toán: Cây quyết định (Decision Tree), Rừng ngẫu nhiên (Random Forest), KNN (K-Nearest Neighbors). Từ đó, xây dựng nền tảng cho các nghiên cứu ứng dụng thực tế vào dự báo giá vé máy bay của các hãng hàng không. Ở chương này, nhóm cũng đã giới thiệu về các chỉ số để đánh giá các mô hình.

1.1. Tổng quan về thuật toán Cây quyết định (Decision Tree):

1.1.1. Giới thiệu về Cây quyết định (Decision Tree):

Cây quyết định là một thuật toán học có giám sát phi tham số, được sử dụng cho cả nhiệm vụ phân loại và hồi quy. Nó có cấu trúc cây, phân cấp, bao gồm nút gốc, các nhánh, nút bên trong và nút lá. Cây quyết định là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật. Các thuộc tính của đối tượng có thể thuộc các kiểu dữ liệu khác nhau như Nhị phân (Binary), Định danh (Nominal), Thứ tự (Ordinal), Số lượng (Quantitative) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal. Tóm lại, cho dữ liệu về các đối tượng gồm các thuộc tính cùng với lớp (classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các dữ liệu chưa biết.

Cây quyết định cũng là một phương tiện có tính mô tả dành cho việc tính toán các xác suất có điều kiện. Cây quyết định có thể được mô tả như là sự kết hợp của các kỹ thuật toán học và tính toán nhằm hỗ trợ việc mô tả, phân loại và tổng quát hóa một tập dữ liệu cho trước.



Hình 1. 1: Mô hình phân loại các nút (node) trong Decision Tree

Các kiểu cây quyết định:

- Cây hồi quy (Regression tree) ước lượng các hàm giá có giá trị là số thực thay vì được sử dụng cho các nhiệm vụ phân loại. (ví dụ: ước tính giá một ngôi nhà hoặc khoảng thời gian một bệnh nhân nằm viện).
- Cây phân loại (Classification tree), nếu y là một biến phân loại như: giới tính (nam hay nữ), kết quả của một trận đấu (thắng hay thua).

1.1.2. Thuật toán Cây quyết định hoạt động như thế nào?

Quyết định về chiến lược phân chia ảnh hưởng rất nhiều đến độ chính xác của cây. Các tiêu chí quyết định là khác nhau đối với cây phân loại và cây hồi quy.

Cây quyết định sử dụng nhiều thuật toán để quyết định chia một nút thành hai hoặc nhiều nút phụ. Việc tạo các nút phụ làm tăng tính đồng nhất của các nút phụ kết quả. Nói cách khác, chúng ta có thể nói rằng độ tinh khiết của nút tăng lên đối với biến mục tiêu. Cây quyết định phân tách các nút trên tất cả các biến có sẵn và sau đó chọn cách phân tách dẫn đến các nút con đồng nhất.

Việc lựa chọn thuật toán cũng dựa trên loại biến mục tiêu. Nhóm sẽ xem xét một số

thuật toán được sử dụng trong Cây quyết định dưới đây:

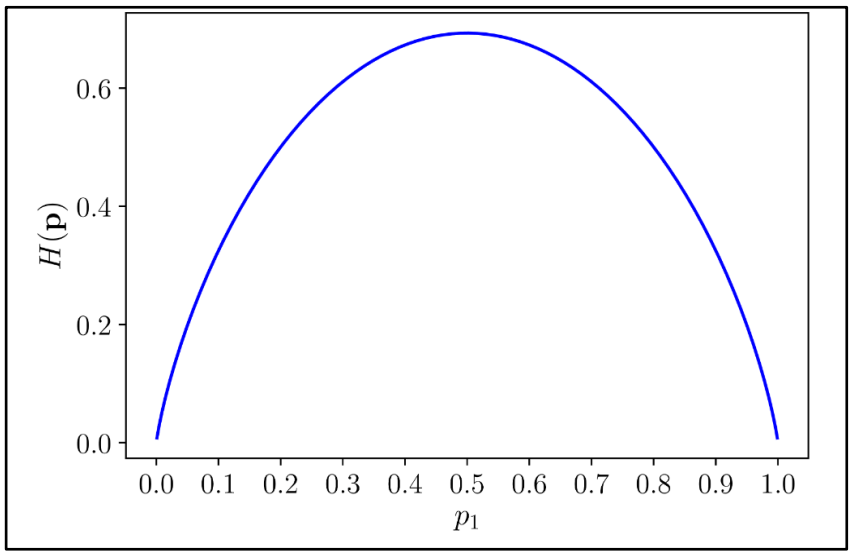
1.1.3. Thuật toán ID3:

Thuật toán ID3 (J. R. Quinlan 1993) một thuật toán Cây quyết định được áp dụng cho các bài toán classification mà tất cả các thuộc tính đều ở dạng categorical. Trong ID3, cần xác định thứ tự của thuộc tính cần được xem xét tại mỗi bước. Thuật toán này sử dụng phương pháp greedy (tham lam) tìm kiếm từ trên xuống thông qua không gian của các nhánh có thể không có backtracking. ID3 sử dụng **hàm số Entropy** và **Information Gain** để xây dựng một cây quyết định.

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

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

Trong đó: log là logarit tự nhiên (một số tài liệu dùng logarit cơ số 2, nhưng giá trị của H(p) chỉ khác đi bằng cách nhân với một hằng số) và quy ước 0 log(0) = 0.



Hình 1. 2: Đồ thị của hàm entropy với n = 2

Xét một ví dụ với n = 2 được cho trên đồ thị. Trong trường hợp p là tinh khiết nhất, tức một trong hai giá trị p; bằng 1, giá trị kia bằng 0, entropy của phân phối này là H(p) = 0. Khi p là vẩn đục nhất, tức cả hai giá trị pi = 0.5, hàm entropy đạt giá trị cao

nhất.

Tổng quát lên với n > 2, hàm entropy đạt giá trị nhỏ nhất nếu có một giá trị p; = 1, đạt giá trị lớn nhất nếu tất cả các p; bằng nhau ((việc này có thể được chứng minh bằng phương pháp nhân tử Lagrange). Những tính chất này của hàm entropy khiến nó được sử dụng trong việc đo độ vẩn đục của một phép phân chia của ID3. Vì lý do này, ID3 còn được gọi là entropy-based decision tree.

Information Gain: dựa trên độ giảm dần của hàm Entropy khi tập dữ liệu được phân chia trên một thuộc tính. Xây dựng một cây quyết định, ta phải tìm tất cả thuộc tính trả về Information Gain cao nhất. Để xác định các nút trong mô hình cây quyết định, ta thực hiện tính Information Gain tại mỗi nút theo trình tự sau:

Bước 1: Tính toán hệ số Entropy của biến mục tiêu S có N phần tử với N_c phần tử thuộc lớp c cho trước.

Bước 2: Tính hàm số Entropy tại mỗi thuộc tính: với thuộc tính x, các điểm dữ liệu trong S được chia ra K nút con S_1, S_2, …, S_K với số điểm trong mỗi nút con lần lượt là m_1, m_2 ,…, m_K.

Bước 3: Chỉ số Gain Information được tính bằng:

$$G(x, S) = H(S) – H(x,S)$$

Các bước hoạt động trong thuật toán ID3:

Bước 1: Thuật toán sẽ bắt đầu chạy với tập dữ liệu ban đầu S hình thành nút gốc.

Bước 2: Trên mỗi lần lặp của thuật toán, nó sẽ tính toán Entropy và Information Gain của thuộc tính này.

Bước 3: Sau đó, thuật toán này sẽ chọn thuộc tính có mức tăng Entropy nhỏ nhất hoặc mức tăng Information Gain lớn nhất.

Bước 4: Tập hợp S sau đó được chia theo thuộc tính đã chọn để tạo ra một tập hợp con của dữ liệu.

Bước 5: Thuật toán tiếp tục lặp lại trên mỗi tập hợp con và chỉ xem xét các thuộc tính chưa từng được chọn trước đó.

1.1.4. Thuật toán CART:

Cây phân loại và hồi quy (CART) là một kĩ thuật học máy có giám sát phổ biến được áp dụng để dự đoán biến mục tiêu định tính (categorical target variable), tạo cây phân loại hoặc biến mục tiêu liên tục (continuous target variable), tạo ra cây hồi quy. Việc phân loại của CART đòi hỏi một cây nhị phân: sự kết hợp của một nút gốc ban đầu, các nút quyết định và các nút cuối. Nút gốc và mỗi nút quyết định đại diện cho một đặc tính và giá trị ngưỡng của đặc tính đó. Trong thuật toán CART, công thức chỉ số Gini được sử dụng để phân loại đối tượng dữ liệu.

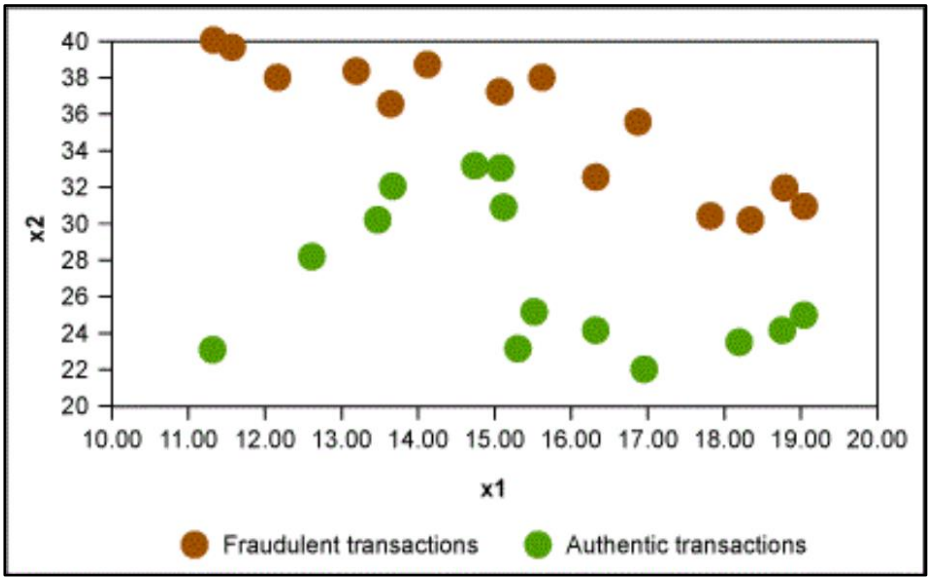
Chỉ số Gini: dùng để xác định mức độ phân tách của cây quyết định. Nó là một chỉ số dùng để đo và đánh giá việc phân chia ở node điều kiện có tốt hay không. Nó dùng để tính độ lệch Gini của nốt cha với tổng các giá trị Gini có đánh trọng số của các node con. Nó dựa vào việc bình phương các xác suất thành viên cho mỗi thể loại đích trong nút. Giá trị của nó tiến đến cực tiểu (bằng 0) khi mọi trường hợp trong nút rơi vào một thể loại đích duy nhất.

Công thức tính chỉ số Gini:

$$I_G(\mathrm{i}) = 1 - \left(\sum_{j=1}^{m} f(i,j)^2\right)$$

Giả sử y nhận các giá trị trong {1, 2,..., m} và gọi f(i,j) là tần suất của giá trị j trong nút i. Nghĩa là f(i,j) là tỷ lệ các bản ghi với y=j được xếp vào nhóm i.

Cách thức hoạt động của thuật toán CART: Giả sử có một tập hợp các giao dịch thẻ tín dụng được dán nhãn là gian lận hoặc xác thực. Có hai thuộc tính của mỗi giao dịch: số tiền (giao dịch) và độ tuổi của khách hàng. Hình bên dưới hiển thị một bản đồ ví dụ về các giao dịch gian lận và xác thực.

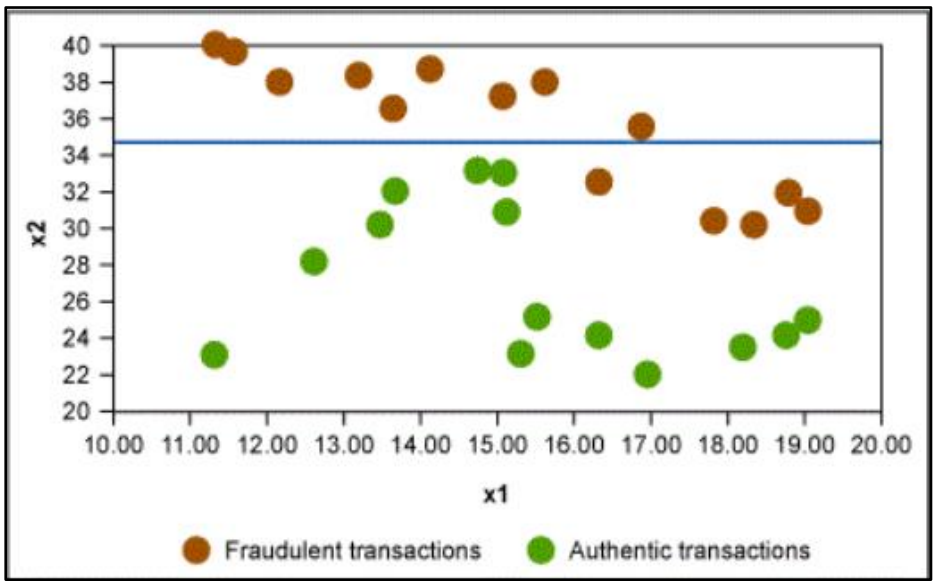


Hình 1. 3: Giao dịch gian lận và xác thực

Thuật toán CART hoạt động để tìm biến độc lập tạo ra nhóm đồng nhất tốt nhất khi tách dữ liệu.

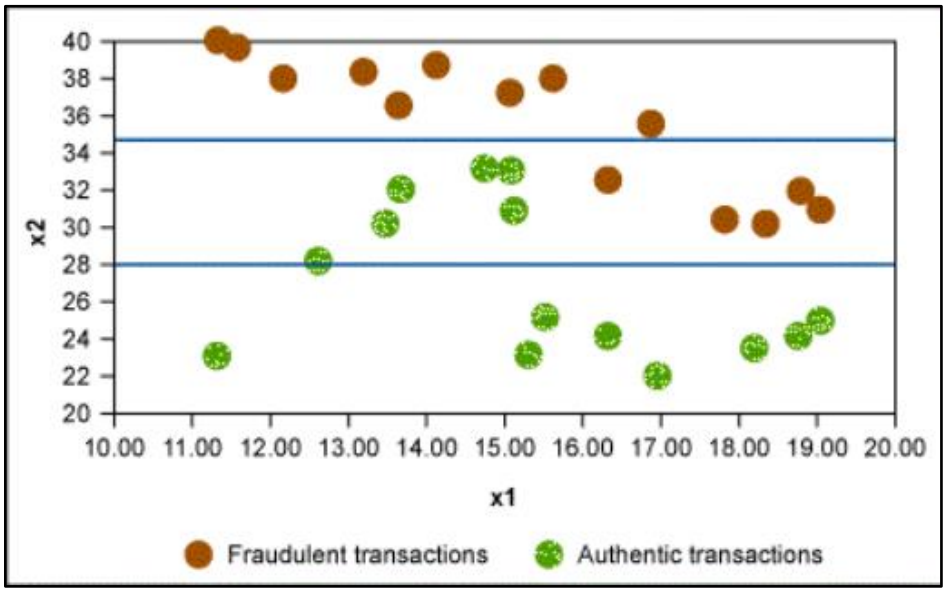
Đối với bài toán hồi quy, tính đồng nhất được đo lường bằng các thống kê như độ lệch chuẩn hoặc phương sai.

Hai tham số quan trọng của kỹ thuật CART là tiêu chí phân tách tối thiểu và tham số phức tạp (C_p). Tiêu chí phân tách tối thiểu là số lượng bản ghi tối thiểu phải có trong một nút trước khi có thể thử phân tách. Điều này phải được chỉ định ngay từ đầu. C_p là một tham số phức tạp để tránh chia nhỏ những nút rõ ràng là không đáng giá. Một cách khác để xem xét các tham số này là C_p giá trị được xác định sau khi "trồng cây" và giá trị tối ưu được sử dụng để "cắt tỉa cây".

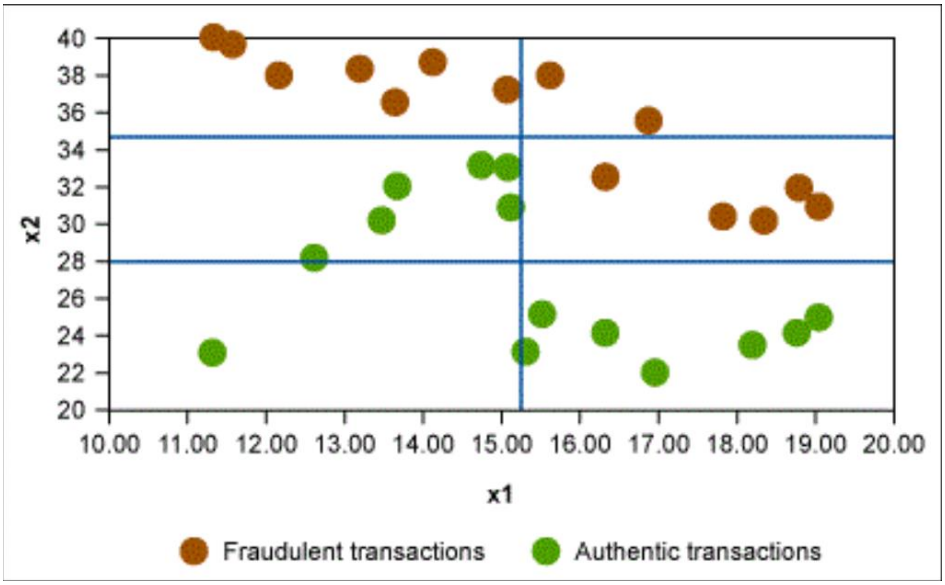


Hình 1. 4: Tách các giao dịch

Trong ví dụ này, cho thấy quy tắc đầu tiên được hình thành là x2 > 35 → giao dịch gian lận. Tương tự, các quy tắc khác được hình thành như trong hai hình bên dưới đây.

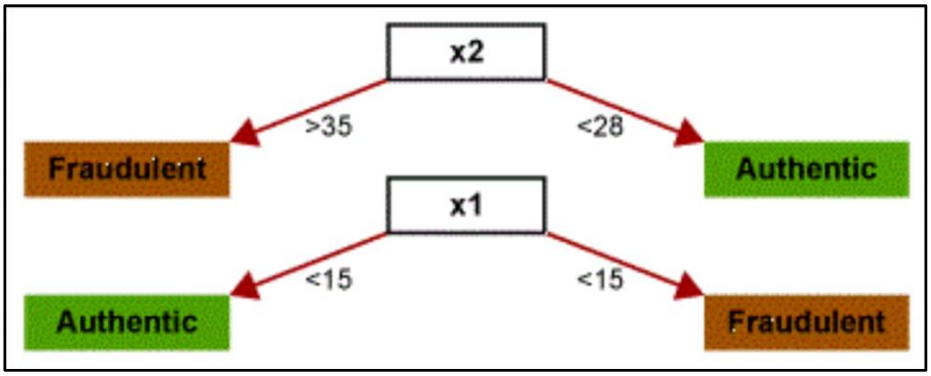


Hình 1. 5: Hai lần tách



Hình 1. 6: Ba lần tách

Bằng cách này, thuật toán CART tiếp tục phân chia tập dữ liệu cho đến khi mỗi nút "lá" còn lại với số lượng bản ghi tối thiểu như được chỉ định bởi tiêu chí phân chia tối thiểu. Điều này dẫn đến một cấu trúc giống cây như thể hiện trong kết quả phân tách. C_p giá trị sau đó được lập biểu đồ dựa trên các cấp độ khác nhau của cây và giá trị tối ưu được sử dụng để cắt tỉa cây.



Hình 1. 7: Kết quả phân tách theo thuật toán CART

1.1.5. Ưu điểm và nhược điểm của thuật toán Cây quyết định:

- **Ưu điểm:** Cây quyết định là một thuật toán đơn giản và phổ biến. Thuật toán này được sử dụng rộng rãi bởi những lợi ích của nó:

+ Mô hình sinh ra các quy tắc dễ hiểu cho người đọc, tạo ra bộ luật với mỗi nhánh lá là một luật của cây.
+ Dữ liệu đầu vào có thể là là dữ liệu missing, không cần chuẩn hóa hoặc tạo biến giả.
+ Có thể xác thực mô hình bằng cách sử dụng các kiểm tra thống kê.
+ Có thể làm việc với cả dữ liệu số và dữ liệu phân loại.
+ Có khả năng là việc với dữ liệu lớn.

- **Nhược điểm:**
  + Thuật toán cây quyết định phụ thuộc rất lớn vào dữ liệu ban đầu. Thậm chí, với một sự thay đổi nhỏ trong bộ dữ liệu, cấu trúc mô hình cây quyết định có thể thay đổi hoàn toàn.
  + Cây quyết định hay gặp vấn đề overfitting.
  + Tốn kém hơn các thuật toán khác: cây quyết định sử dụng phương pháp greedy (tìm kiếm tham lam) trong quá trình xây dựng, chúng có thể tốn kém hơn để đào tạo so với các thuật toán khác.

1.2. Tổng quan về thuật toán Random Forest

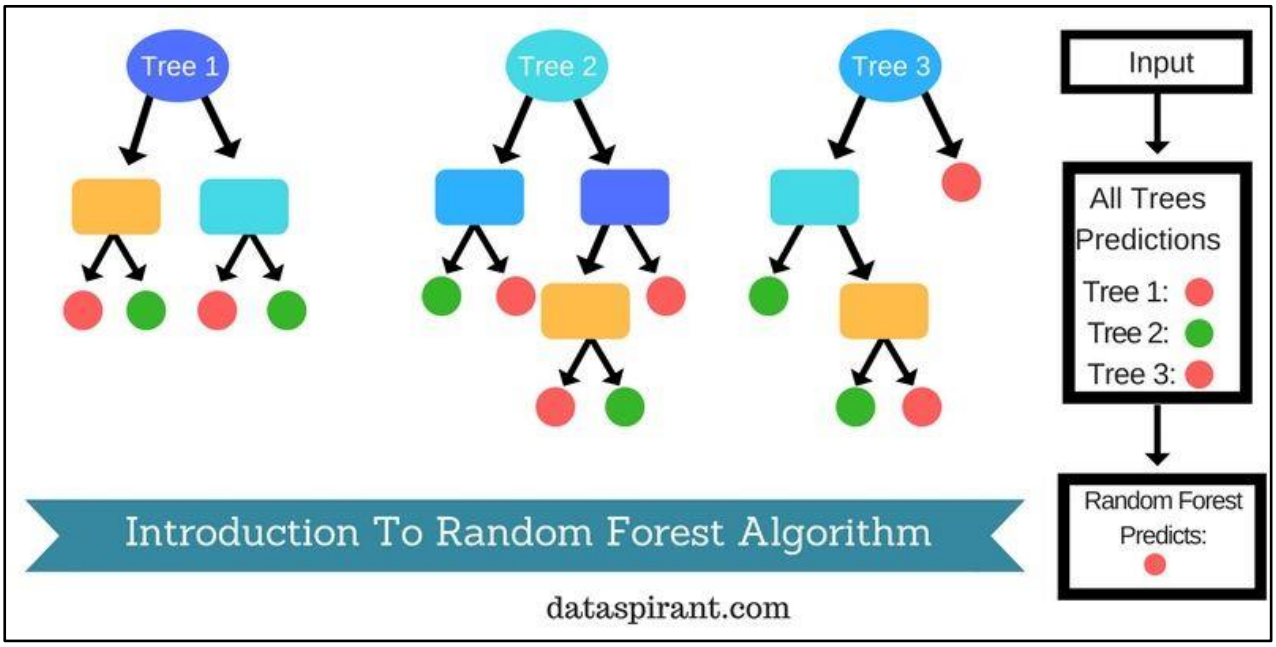
1.2.1. Giới thiệu thuật toán:

Thuật ngữ Random Forest được xuất hiện lần đầu tiên vào năm 1995, sau đó kết hợp với kỹ thuật lựa chọn các thuộc tính ngẫu nhiên của Leo Breiman năm 1996. Năm 2001, Leo Breiman xây dựng thuật toán Random Forest có bổ sung thêm một lớp ngẫu nhiên để phân lớp. Ngoài việc mỗi cây sử dụng các mẫu dữ liệu khác nhau, rừng ngẫu nhiên được thay đổi để xây dựng các cây phân loại và hồi quy khác nhau. Các gói thư viện cài đặt thuật toán Random Forest đầu tiên được xây dựng bằng ngôn ngữ Fortran bởi Leo Breiman và Cutler. Random Forest được mô hình hóa như tập các cây phân lớp. Tuy nhiên Random Forest sử dụng các mẫu ngẫu nhiên cho các cây cũng như việc chọn lựa thuộc tính ngẫu nhiên khi phân chia cây. Thuật toán Random Forest tỏ ra chính xác và nhanh hơn khi huấn luyện trên không gian dữ liệu lớn với nhiều thuộc tính, việc sử dụng kết quả dự đoán của cả tất cả các cây trong

rừng khi phân lớp hoặc hồi quy giúp cho kết quả thuật toán chính xác hơn.

Random Forest có thể giải quyết cả bài toán hồi quy (regression) và phân loại (classification). Nó cũng là thuật toán linh hoạt và dễ sử dụng nhất. Như tên gọi của nó, Random Forest (RF) dựa trên cơ sở:

- Random: tính ngẫu nhiên.
- Forest: nhiều cây quyết định (decision tree).

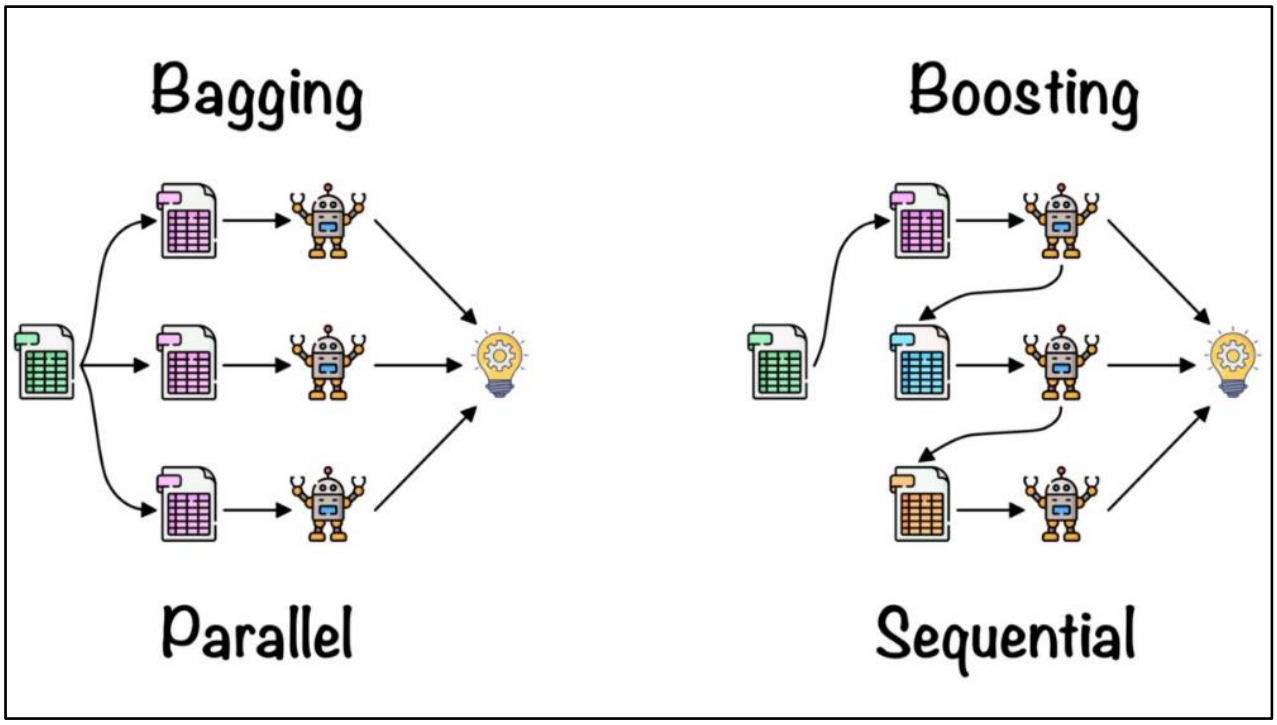


Hình 1. 8: Thuật toán Rừng ngẫu nhiên (Random Forest)

Random Forest tạo ra cây quyết định trên các mẫu dữ liệu được chọn ngẫu nhiên, được dự đoán từ mỗi cây và chọn giải pháp tốt nhất bằng cách bỏ phiếu (voting). Nó cũng cung cấp một chỉ báo khá tốt về tầm quan trọng của tính năng. Random Forests có nhiều ứng dụng, chẳng hạn như công cụ đề xuất, phân loại hình ảnh và lựa chọn tính năng. Nó có thể được sử dụng để phân loại các ứng viên cho vay trung thành, xác định hoạt động gian lận và dự đoán các bệnh.

Trước khi tìm hiểu hoạt động của RF, chúng ta phải xem xét về phương pháp kết hợp (phương pháp Ensemble). Có thể sử dụng để tăng độ chính xác trên tập dữ liệu là kết hợp một số mô hình với nhau. Phương pháp này gọi là esemble. Phương pháp Ensemble Learning được chia thành những loại sau đây:

- Bagging (kỹ thuật đóng gói): xây dựng một lượng lớn các mô hình (thường là cùng loại) trên những mẫu con (subsamples) khác nhau từ tập huấn luyện dữ liệu một cách song song nhằm đưa ra dự đoán tốt hơn.
- Boosting (kỹ thuật tăng cường): xây dựng một lượng lớn các mô hình (thường là cùng loại). Tuy nhiên quá trình huấn luyện trong phương pháp này diễn ra tuần tự theo chuỗi (sequence). Trong chuỗi này mỗi mô hình sau sẽ học cách sửa những lỗi của dữ liệu trong các mô hình trước. Cụ thể là mục tiêu của kỹ thuật tăng cường này sẽ khắc phục và giải quyết các dữ liệu mà mô hình trước dự đoán sai nhằm đưa ra dự đoán tốt hơn trong thuật toán.



Hình 1. 9: Hai kỹ thuật đặc trưng trong phương pháp Ensemble

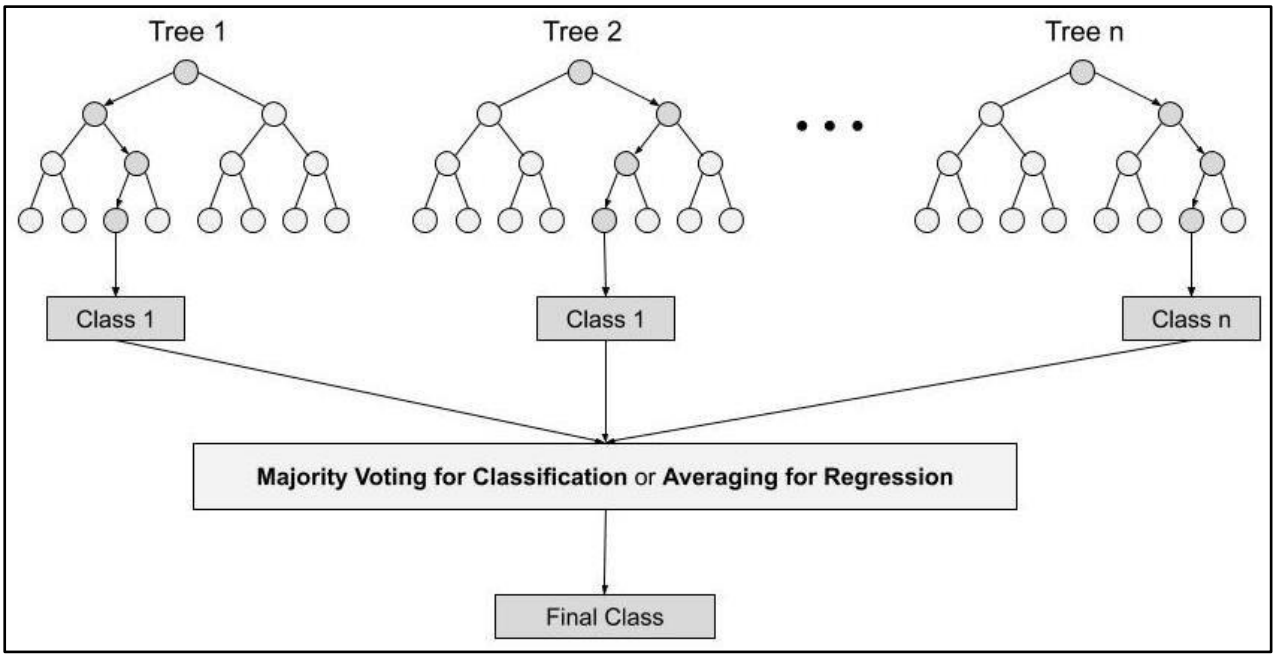
1.2.2. Cách thức hoạt động của thuật toán Random Forest (RF):

1.2.2.1. Kỹ thuật đóng gói (Bagging) trong RF hoạt động theo các bước sau:

- Bước 1: Chọn các mẫu ngẫu nhiên từ tập dữ liệu đã cho (chọn điểm dữ liệu "k" ngẫu nhiên từ tập huấn luyện "m", lưu ý k < m). Từ tập "k" thuộc tính, tính toán ra node "d" là tốt nhất cho node phân loại. Chia các node con theo

node tốt nhất vừa tìm được. Lặp lại các thao tác cho đến khi đạt đến k node để tạo ra "n" cây quyết định.

- Bước 2: Thiết lập cây quyết định cho từng mẫu và nhận kết quả dự đoán từ mỗi quyết định cây.
- Bước 3: Bỏ phiếu cho mỗi kết quả dự đoán. Tính toán số lượng phiếu bầu trên toàn bộ Forest cho từng kết quả.
- Bước 4: Chọn kết quả được xem xét dựa trên bỏ phiếu đa số (được dự đoán nhiều nhất là dự đoán cuối cùng) hoặc trung bình để phân loại và hồi quy tương ứng.



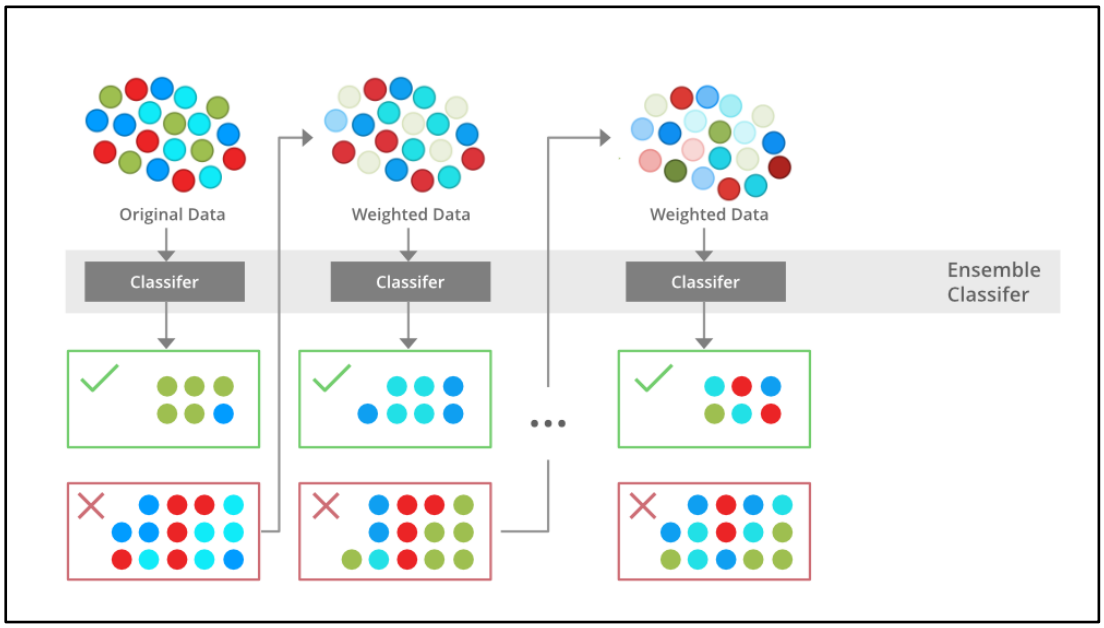
Hình 1. 10: Các bước hoạt động của Bagging trong thuật toán Random Forest

1.2.2.2. Kỹ thuật tăng cường (Boosting) trong RF hoạt động theo các bước sau:

- Bước 1: Khởi tạo tập dữ liệu và gán trọng số bằng nhau cho điểm dữ liệu.
- Bước 2: Cung cấp dữ liệu này làm đầu vào cho mô hình và xác định các điểm dữ liệu được phân loại sai.
- Bước 3: Tăng trọng số của các điểm dữ liệu được phân loại sai và giảm trọng số của các điểm dữ liệu được phân loại chính xác. Và sau đó chuẩn hóa trọng

số của tất cả các điểm dữ liệu.

- Bước 4: Đưa ra kết quả dự đoán sau khi đã chuẩn hóa dữ liệu để tiến hành phân loại và hồi quy tương ứng.



Hình 1. 11: Minh họa về quá trình huấn luyện trong kỹ thuật tăng cường diễn ra tuần tự theo chuỗi

1.2.3. Vấn đề tối ưu tham số trong Random Forest:

Đối với các thuật toán nói chung và Random Forest nói riêng, việc tối ưu các tham số là quan trọng và cần thiết. Khi thay đổi giá trị hoặc cách tính giá trị một số tham số trong thuật toán có thể làm thay đổi độ chính xác, thời gian thực thi cũng như tài nguyên hệ thống của thuật toán. Random Forest cũng có một số tham số khi huấn luyện, vậy nên việc lựa chọn tham số sao cho thuật toán đạt hiệu quả cao và tốt nhất là rất cần thiết khi huấn luyện cũng như đưa vào ứng dụng cụ thể.

Các siêu tham số tối ưu quan trọng trong thuật toán: Các siêu tham số trong rừng ngẫu nhiên được sử dụng để tăng khả năng dự đoán của mô hình hoặc để làm cho mô hình nhanh hơn:

- Tăng sức mạnh dự đoán:
    + Siêu tham số n_estimators, chỉ là số cây mà thuật toán xây dựng trước

khi lấy phiếu bầu tối đa hoặc lấy giá trị trung bình của các dự đoán. Nói chung, số lượng cây cao hơn làm tăng hiệu suất và làm cho các dự đoán ổn định hơn, nhưng nó cũng làm chậm quá trình tính toán.

+ Một siêu tham số quan trọng khác là max_features, là số lượng tối đa các thuộc tính mà RF được phép thử trong từng cây riêng lẻ. Việc tăng max_features thường cải thiện hiệu suất của mô hình vì tại mỗi nút hiện tại sẽ có nhiều tùy chọn hơn để xem xét. Tuy nhiên, việc này không nhất thiết phải làm vì nó làm giảm tính đa dạng của từng cây vốn là đặc điểm riêng biệt (USP) của rừng ngẫu nhiên. Do đó, cần đạt được sự cân bằng phù hợp và chọn max_features tối ưu.
+ Siêu tham số quan trọng cuối cùng là min_sample_leaf. Điều này xác định số lượng lá tối thiểu cần thiết để tách một nút bên trong.

- Tăng tốc độ của mô hình:
  + Siêu tham số n_jobs cho động cơ biết nó được phép sử dụng bao nhiêu bộ xử lý. Nếu nó có giá trị là một, nó chỉ có thể sử dụng một bộ xử lý. Giá trị "-1" có nghĩa là không có giới hạn.
  + Siêu tham số random_state làm cho đầu ra của mô hình có thể sao chép được. Mô hình sẽ luôn tạo ra cùng một kết quả khi nó có một giá trị xác định là random_state, nó được cung cấp cùng một siêu tham số và cùng một dữ liệu huấn luyện.
  + Cuối cùng, có oob_score (còn được gọi là lấy mẫu oob), là một phương pháp xác thực chéo rừng ngẫu nhiên. Trong lần lấy mẫu này, khoảng một phần ba dữ liệu không được sử dụng để đào tạo mô hình và có thể được sử dụng để đánh giá hoạt động của nó. Những mẫu này được gọi là mẫu xuất túi. Nó rất giống với việc để lại một kỹ thuật xác nhận. Phương pháp này chỉ đơn giản là gắn thẻ mọi quan sát được sử dụng trong cây khác nhau. Và sau đó, nó tìm ra số phiếu tối đa cho mọi quan sát.

1.2.4. Ưu điểm và nhược điểm của thuật toán Random Forest:

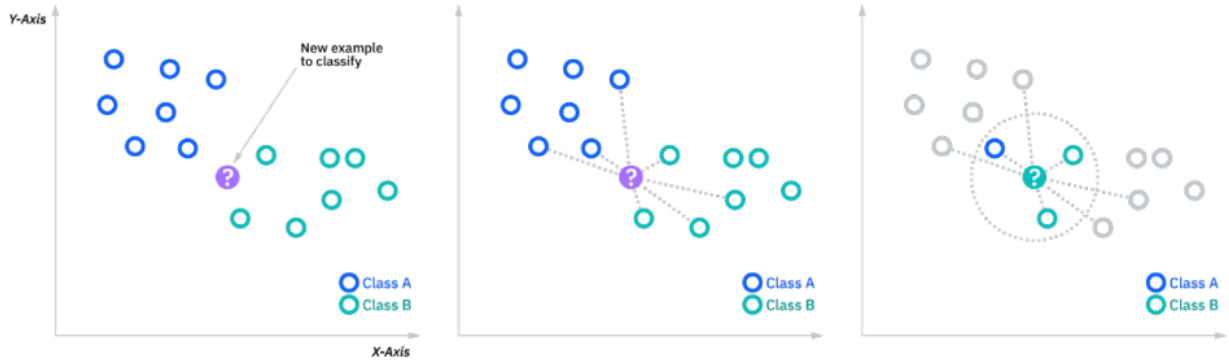
- **Ưu điểm:**
    + Random Forest có khả năng thực hiện cả hai nhiệm vụ Phân loại và Hồi quy.
    + Có khả năng xử lý các tập dữ liệu lớn.
    + Nâng cao độ chính xác của mô hình và ngăn chặn mô hình quá khớp với dữ liệu (overfitting).
    + Nó hoạt động tốt ngay cả khi dữ liệu chứa các giá trị null/bị thiếu.
    + Thuật toán chạy rất ổn định vì các kết quả trung bình được đưa ra bởi một số lượng lớn cây được lấy.
- **Nhược điểm:**
    + Mặc dù rừng ngẫu nhiên có thể được sử dụng cho cả nhiệm vụ phân loại và hồi quy, nó không phù hợp hơn cho các nhiệm vụ hồi quy.
    + Mất nhiều thời gian đào tạo hơn so với các mô hình khác do sự phức tạp của nó. Bất cứ khi nào nó phải đưa ra một dự đoán, mỗi cây quyết định phải tạo ra đầu ra cho dữ liệu đầu vào đã cho.

1.3. Tổng quan về thuật toán KNN (K-Nearest Neighbors)

1.3.1. Giới thiệu thuật toán:

Thuật toán phân loại KNN được phát triển từ nhu cầu phân tích phân biệt khi các ước tính tham số đáng tin cậy về mật độ xác suất không xác định hoặc khó xác định. Trong một báo cáo chưa được công bố của Trường Y khoa Hàng không Hoa Kỳ vào năm 1951, Fix và Hodges đã giới thiệu một phương pháp phi tham số để phân loại mẫu và từ đó đã được biết đến quy tắc k-nearest neighbor. Họ đã giới thiệu một cách tiếp cận mới để phân loại phi tham số bằng cách dựa vào 'khoảng cách' giữa các điểm hoặc phân phối. Ý tưởng cơ bản là phân loại một cá nhân vào quần thể có mẫu chứa phần lớn 'nearest neighbor'. Sau đó vào năm 1967, một số thuộc tính chính thức của quy tắc k-nearest neighbor đã được tìm ra các rủi ro của KNN. Khi các thuộc tính chính thức của phân loại KNN được thiết lập, một chuỗi các nghiên cứu sau đó bao gồm các phương pháp

rejection approaches, các sàng lọc liên quan đến Bayes error rate, phương pháp tiếp cận theo trọng số khoảng cách và phương pháp soft computing.

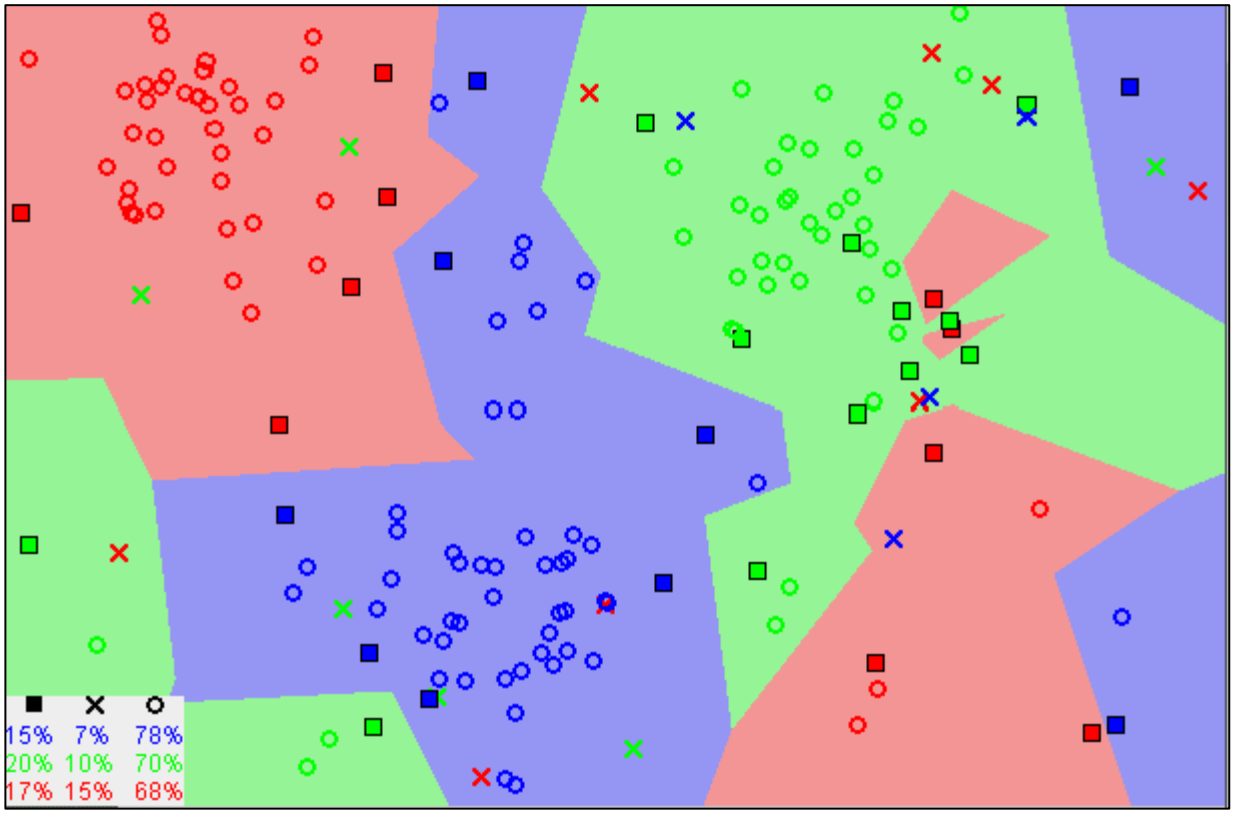


Hình 1. 12: Minh họa về thuật toán KNN

1.3.2. Cách thức hoạt động:

Với KNN, trong bài toán Classification, label của một điểm dữ liệu mới (hay kết quả của câu hỏi trong bài thi) được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong training set. Label của một test data có thể được quyết định bằng major voting (bầu chọn theo số phiếu) giữa các điểm gần nhất, hoặc nó có thể được suy ra bằng cách đánh trọng số khác nhau cho mỗi trong các điểm gần nhất đó rồi suy ra label.

Trong bài toán Regression, đầu ra của một điểm dữ liệu sẽ bằng chính đầu ra của điểm dữ liệu đã biết gần nhất (trong trường hợp K=1), hoặc là trung bình có trọng số của đầu ra của những điểm gần nhất, hoặc bằng một mối quan hệ dựa trên khoảng cách tới các điểm gần nhất đó.



Hình 1. 13: Biểu đồ mô tả sự gần nhau của các điểm dữ liệu

Các bước thực hiện KNN:

- Bước 1: Xác định tập dữ liệu đã gán nhãn và chưa gán nhãn.
- Bước 2: Tính khoảng cách giữa điểm dữ liệu chưa gán nhãn và dữ liệu đã gắn nhãn theo các nhóm của tập dữ liệu.
- Bước 3: Sắp xếp các khoảng cách và chỉ số theo thứ tự từ nhỏ nhất đến lớn nhất (theo thứ tự tăng dần) theo khoảng cách.
- Bước 4: Chọn K khoảng cách nhỏ nhất.
- Bước 5: Kiểm tra danh sách các lớp có khoảng cách ngắn nhất và đếm số lượng của mỗi lớp xuất hiện.
- Bước 6: Lấy nhãn của K mục đã chọn.
- Bước 7: Nếu hồi quy, trả về mean của nhãn K.
- Bước 8: Nếu phân loại thì trả về mode của nhãn K.

Các cách chọn K:

Giá trị k trong thuật toán KNN xác định số lượng nhóm sẽ được kiểm tra để xác định phân loại của một điểm truy vấn cụ thể. Ví dụ: nếu k=1, cá thể sẽ được gán vào cùng một lớp với nhóm gần nhất của nó. Việc xác định k có thể là một hành động cân bằng vì các giá trị khác nhau có thể dẫn đến trang bị thừa hoặc thiếu. Các giá trị k thấp hơn có thể có phương sai cao, nhưng độ lệch thấp và các giá trị k lớn hơn có thể dẫn đến độ lệch cao và phương sai thấp hơn. Việc lựa chọn k phần lớn sẽ phụ thuộc vào dữ liệu đầu vào vì dữ liệu có nhiều ngoại lệ hoặc nhiễu hơn sẽ có khả năng hoạt động tốt hơn với giá trị k cao hơn. Nên có một số lẻ cho k để tránh ràng buộc trong phân loại và các chiến thuật xác thực chéo.

Cách tính khoảng cách:

- Khoảng cách euclidean: Đây là thước đo khoảng cách được sử dụng **phổ biến nhất** và được giới hạn ở các vectơ có giá trị thực. Kết quả là đo một đường thẳng giữa điểm truy vấn và điểm khác được đo.

$$d\,(x,y)\;=\;\sqrt{\sum_{i=1}^{n}\,(y_i-x_i)^2}$$

- Khoảng cách Manhattan: Đây cũng là một thước đo khoảng cách phổ biến khác, đo giá trị tuyệt đối giữa hai điểm.

$$Manhattan\ Distance\;=\;d(x,y)\;=\left(\sum_{i=1}^{m}\,|x_i-y_i|\right)$$

- Khoảng cách Minkowski: Thước đo khoảng cách này là dạng tổng quát của thước đo khoảng cách Euclide và Manhattan. Tham số, p, cho phép tạo các số liệu khoảng cách khác. Khoảng cách Euclide được biểu thị bằng công thức này khi p bằng hai và khoảng cách Manhattan được biểu thị bằng p bằng một.

$$Minkowski\ Distance\;=\left(\sum_{i=1}^{n}\,|x_i-y_i|\right)^{\frac{1}{p}}$$

- Khoảng cách Hamming: Kỹ thuật này thường được sử dụng với các vectơ Boolean hoặc chuỗi, xác định các điểm mà các vectơ không khớp. Do đó, nó còn được gọi là chỉ số trùng lặp.

$$Hamming\ Distance\ =\ D_H = \left(\sum_{i=1}^{k} |x_i - y_i|\right)$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D \neq 1$$

1.3.3. Ưu điểm và nhược điểm của thuật toán:

- **Ưu điểm:**

+ Thuật toán đơn giản, dễ thực hiện và trực quan.
+ Không cần xây dựng mô hình, điều chỉnh một số tham số hoặc đưa ra các giả định bổ sung.
+ Thuật toán rất linh hoạt. Nó có thể được sử dụng để phân loại, hồi quy.

- **Nhược điểm:** Mặc dù có những ưu điểm nêu trên, KNN có một vài hạn chế:

+ KNN có thể có hiệu suất thời gian chạy kém khi tập huấn luyện lớn. Nó rất nhạy cảm với các đặc điểm không liên quan hoặc dư thừa bởi vì tất cả các đặc điểm đều góp phần tạo nên sự giống nhau và do đó góp phần vào việc phân loại.
+ Thuật toán học dựa trên khoảng cách chưa được rõ ràng nên sử dụng loại khoảng cách nào và sử dụng thuộc tính nào để tạo ra kết quả tốt nhất.
+ Chi phí tính toán khá cao vì chúng ta cần tính toán khoảng cách của mỗi phiên bản truy vấn tới tất cả các mẫu huấn luyện.

1.4. Các chỉ số đánh giá:

Để đánh giá các mô hình, nhóm lựa chọn các thông số MAE, MSE, RMSE, R2 Score, RMSLE, MAPE, Adjusted R Square.

1.4.1. Mean Absolute Error (MAE):

Giá trị sai số tuyệt đối trung bình (MAE) là thước đo cho thấy độ chính xác giữa các giá trị dự đoán so với giá trị thực tế. MAE tính trung bình của tổng các giá trị tuyệt đối các sai số. Công thức MAE được định nghĩa như sau:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

1.4.2. Mean Squared Error (MSE):

Sai số bình phương trung bình (MSE) là giá trị trung bình của chênh lệch bình phương giữa giá trị dự đoán và giá trị quan sát được. MSE là thước đo chất lượng của mô hình hồi quy tuyến tính - nó luôn không âm và các giá trị càng gần 0 càng tốt. Công thức tính MSE:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)$$

Trong đó:

- y_i là giá trị quan sát
- $\hat{y}_i$ là giá trị dự đoán
- N là tổng số quan sát

Chỉ số MSE càng nhỏ chứng tỏ sự chênh lệch giữa giá trị dự đoán và giá trị quan sát càng bé thì mô hình sẽ dự báo càng có tính chính xác cao.

1.4.3. Root Mean Squared Error (RMSE):

Sai số bình phương trung bình gốc (RMSE) là một thước đo được sử dụng để đánh giá sự khác biệt giữa các giá trị dự đoán và giá trị thực tế. RMSE được định nghĩa là căn bậc hai của sai số bình phương trung bình.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

RMSE càng nhỏ tức là sai số càng bé thì mức độ ước lượng cho thấy độ tin cậy của mô hình có thể đạt cao nhất.

1.4.4. R-Square:

Hệ số xác định R bình phương (R Square) cho biết mức độ phù hợp của mô hình nghiên cứu với các biến ở mức bao nhiêu %.

Giá trị R square nằm trong khoảng từ 0 đến 1 và giá trị này càng lớn càng cho thấy sự mô hình dự đoán càng tốt.

$$\boldsymbol{R^2} = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

1.4.5. Root Mean Squared Logarithmic Error (RMSLE):

Sai số bình phương trung bình gốc của các giá trị dự đoán và thực tế được chuyển đổi logarit. RMSLE thêm 1 vào cả hai giá trị thực tế và dự đoán trước khi lấy logarit tự nhiên để tránh logarit của các giá trị 0 (không) có thể. Do đó, hàm có thể được sử dụng nếu thực tế hoặc dự đoán có các phần tử có giá trị bằng không.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log y_i - \log \hat{y}_i)^2}$$

1.4.6. Mean Absolute Percentage Error (MAPE):

Trung bình độ lệch tương đối MAPE tính đến độ lớn tương đối của độ lệch dự báo so với độ lớn giá trị thực, mô hình có Trung bình độ lệch tương đối MAPE càng nhỏ thì dự báo càng chính xác.

$$MAPE = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

Trong đó:

- A_t là giá trị thực tế
- F_t là giá trị dự báo
- n là tổng số lượng điểm
- t là thời điểm dự báo

1.4.7. Adjusted R Square:

Ý nghĩa của R bình phương hiệu chỉnh cũng giống như R bình phương là phản ánh mức độ phù hợp của mô hình. R bình phương hiệu chỉnh được tính từ R bình phương thường được sử dụng hơn vì giá trị này phản ánh sát hơn mức độ

phù hợp của mô hình hồi quy tuyến tính đa biến. R bình phương hiệu chỉnh không nhất thiết tăng lên khi chúng ta đưa thêm các biến độc lập vào mô hình. Với mô hình hồi quy tuyến tính với quá nhiều biến độc lập, thì chỉ số R bình hiệu chỉnh sẽ phù hợp hơn so với R bình phương vì nó không thổi phồng mức độ phù hợp của mô hình.

1.5. Mã hóa dữ liệu phân loại (Encoding categorical features):

1.5.1. LabelEncoder:

Label Encoding là phương pháp mã hóa bằng cách gắn nhãn, cách tiếp cận này rất đơn giản và nó liên quan đến việc chuyển đổi từng giá trị trong một cột thành một số. Với các giá trị khác nhau trong một trường dữ liệu, chúng sẽ được gán với bấy nhiêu nhãn bằng số thay vì dạng phân loại.

SAFETY-LEVEL (TEXT)	SAFETY-LEVEL (NUMERICAL)
None	0
Low	1
Medium	2
High	3
Very-High	4

Hình 1. 14: Ví dụ về Label Encoding

1.5.2. One-hot Encoding:

One-hot encoding là quá trình chuyển đổi dữ liệu phân loại thành dữ liệu số để sử dụng trong học máy. Các tính năng phân loại được chuyển thành các tính năng nhị phân được mã hóa "one-hot", nghĩa là nếu một tính năng được đại diện bởi cột đó, nó sẽ nhận được 1. Nếu không, nó sẽ nhận được 0.

One-hot encoding (OHC) thường được sử dụng để đánh địa chỉ các bảng tra cứu và trong thiết kế bộ lọc FIR. Đối với các số nằm trong khoảng từ 0-N, cần có N +1 bit. Chỉ một trong số các bit chiếm vị trí '1' và tất cả các bit còn lại đều là số không. '1' đại diện cho giá trị vị trí tương ứng của số đã cho.

Regular representation	One-hot coding
0	00000001
1	00000010
2	00000100
3	00001000
4	00010000
5	00100000
6	01000000
7	10000000

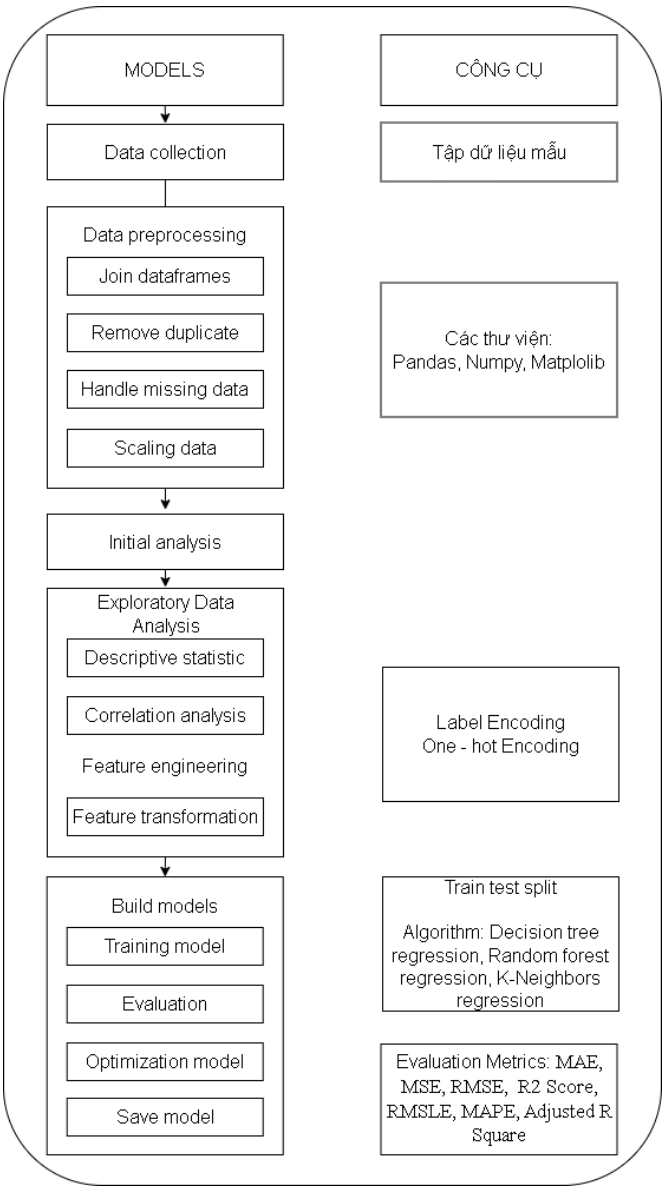
Hình 1. 15: Minh họa về One-hot Coding

CHƯƠNG 2: THU THẬP VÀ TIỀN XỬ LÝ DỮ LIỆU

Tóm tắt chương 2:

Nội dung của chương này trình bày tổng quan về bộ dữ liệu được tìm kiếm trên Kaggle bao gồm các thông tin liên quan đến chuyến bay được trích xuất từ website booking: **Ease my trip** - là một trong những công ty du lịch hàng đầu của Ấn Độ và là một cái tên đáng tin cậy trong ngành du lịch Ấn Độ, mô tả ý nghĩa của dữ liệu thể hiện. Thực hiện tiền xử lý dữ liệu và trực quan hóa dữ liệu EDA.

2.1. Quy trình thực nghiệm:



Hình 2. 1: Mô tả quy trình thực nghiệm

2.2. Thu thập và mô tả dữ liệu:

Bộ dữ liệu được tìm kiếm trên Kaggle bao gồm các thông tin liên quan đến chuyến bay được trích xuất từ website booking: **Ease my trip** - là một trong những công ty du lịch hàng đầu của Ấn Độ và là một cái tên đáng tin cậy trong ngành du lịch Ấn Độ. Nội dung dữ liệu bao gồm thông tin chuyến bay của loại vé phổ thông và thương gia trong vòng 50 ngày (từ ngày 11/02 đến 31/03/2022) của 6 hãng hàng không nổi tiếng tại Ấn Độ. Tập dữ liệu gồm có 300261 dòng và 11 trường dữ liệu:

- Airline: Tên hãng hàng không. Đây là một biến phân loại có 6 hãng hàng không khác nhau (Indigo, Air_Indi, Go_first, AirAsia, Spicejet, Vistara)
- Flight: Mã chuyến bay của máy bay. Là một biến phân loại.
- Source City: Thành phố chuyến bay cất cánh. Đây là một biến phân loại có 6 thành phố khác nhau (Chennai, Delhi, Hyderabad, Kolkata, Mumbai, Bangalore)
- Departure Time: Thời gian khởi hành. Đây là một biến phân loại với 6 nhãn thời gian khác nhau trong ngày.
- Stops: Số điểm dừng giữa điểm đầu và điểm cuối của chuyến bay. Là một biến phân loại với 3 giá trị (zero, one, two or more)
- Arrival Time: Thời gian đến. Đây là một biến phân loại với 6 nhãn thời gian khác nhau trong ngày.
- Destination City: Thành phố chuyến bay hạ cánh. Đây là một biến phân loại có 6 thành phố khác nhau (Chennai, Delhi, Hyderabad, Kolkata, Mumbai, Bangalore)
- Class: Hạng vé. Là một biến phân loại chứa thông tin về hạng ghế; nó có hai giá trị riêng biệt: phổ thông và thương gia.
- Duration: Là một biến liên tục hiển thị tổng thời gian cần thiết để di chuyển giữa các thành phố tính bằng giờ.
- Days Left: Ngày còn lại, được tính bằng cách lấy ngày đặt trước trừ đi ngày của chuyến bay.
- Price: Giá vé.

Các hàng dữ liệu:

	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1	5953
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955

Các cột dữ liệu trên có kiểu dữ liệu như sau:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 93487 entries, 0 to 93486
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ID                93487 non-null  int64
 1   airline           93487 non-null  object
 2   ch_code           93487 non-null  object
 3   num_code          93487 non-null  int64
 4   source_city       93487 non-null  object
 5   departure_time    93487 non-null  object
 6   stops             93487 non-null  object
 7   arrival_time      93487 non-null  object
 8   destination_city  93487 non-null  object
 9   class             93487 non-null  object
 10  duration          93487 non-null  float64
 11  days_left         93487 non-null  int64
 12  price             93487 non-null  int64
dtypes: float64(1), int64(4), object(8)
memory usage: 9.3+ MB
```

=> Từ những thông tin cơ bản về chuyến bay bao gồm giá vé và các thông tin ảnh hưởng đến giá bay của các hãng hàng không, giá vé máy bay sẽ được dự báo và được kiểm tra bằng các chỉ số chính xác.

2.3. Thư viện sử dụng

```
# import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

2.4. Tiền xử lý dữ liệu:

- Đọc dữ liệu:

```
# read data

df_bu = pd.read_csv('business.csv')

df_ec = pd.read_csv('economy.csv')
```

```
[ ] df_bu
```

	ID	airline	ch_code	num_code	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
0	1	Air_India	AI	868	Delhi	Evening	zero	Evening	Mumbai	Business	2.00	1	25612
1	2	Air_India	AI	624	Delhi	Evening	zero	Night	Mumbai	Business	2.25	1	25612
2	3	Air_India	AI	531	Delhi	Evening	one	Night	Mumbai	Business	24.75	1	42220
3	4	Air_India	AI	839	Delhi	Night	one	Night	Mumbai	Business	26.50	1	44450
4	5	Air_India	AI	544	Delhi	Evening	one	Night	Mumbai	Business	6.67	1	46690
...	...	...	...	...	...	...	...	...	...	...	...	...	...
93482	93483	Vistara	UK	822	Chennai	Morning	one	Evening	Hyderabad	Business	10.08	49	69265
93483	93484	Vistara	UK	826	Chennai	Afternoon	one	Night	Hyderabad	Business	10.42	49	77105
93484	93485	Vistara	UK	832	Chennai	Early_Morning	one	Night	Hyderabad	Business	13.83	49	79099
93485	93486	Vistara	UK	828	Chennai	Early_Morning	one	Evening	Hyderabad	Business	10.00	49	81585
93486	93487	Vistara	UK	822	Chennai	Morning	one	Evening	Hyderabad	Business	10.08	49	81585

93487 rows × 13 columns

```
df_ec['class'] = ['Economy']*len(df_ec)

# merge dataframes

df = pd.concat([df_bu, df_ec], ignore_index=True)

df
```

```
[ ] df_ec
```

	ID	airline	ch_code	num_code	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
0	1	SpiceJet	SG	8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1	5953
1	2	SpiceJet	SG	8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953
2	3	AirAsia	I5	764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956
3	4	Vistara	UK	995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955
4	5	Vistara	UK	963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955
...	...	...	...	...	...	...	...	...	...	...	...	...	...
206610	206611	Vistara	UK	832	Chennai	Early_Morning	one	Night	Hyderabad	Economy	13.83	49	7697
206611	206612	Vistara	UK	832	Chennai	Early_Morning	one	Night	Hyderabad	Economy	13.83	49	7709
206612	206613	Vistara	UK	826	Chennai	Afternoon	one	Morning	Hyderabad	Economy	20.58	49	8640
206613	206614	Vistara	UK	822	Chennai	Morning	one	Morning	Hyderabad	Economy	23.33	49	8640
206614	206615	Vistara	UK	824	Chennai	Night	one	Night	Hyderabad	Economy	24.42	49	8640

206615 rows × 13 columns

- Gộp dữ liệu:

```
# insert Class column

df_bu['class'] = ['Business']*len(df_bu)
```

	ID	airline	ch_code	num_code	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
0	1	Air_India	AI	868	Delhi	Evening	zero	Evening	Mumbai	Business	2.00	1	25612
1	2	Air_India	AI	624	Delhi	Evening	zero	Night	Mumbai	Business	2.25	1	25612
2	3	Air_India	AI	531	Delhi	Evening	one	Night	Mumbai	Business	24.75	1	42220
3	4	Air_India	AI	839	Delhi	Night	one	Night	Mumbai	Business	26.50	1	44450
4	5	Air_India	AI	544	Delhi	Evening	one	Night	Mumbai	Business	6.67	1	46690
...	...	...	...	...	...	...	...	...	...	...	...	...	...
300097	206611	Vistara	UK	832	Chennai	Early_Morning	one	Night	Hyderabad	Economy	13.83	49	7697
300098	206612	Vistara	UK	832	Chennai	Early_Morning	one	Night	Hyderabad	Economy	13.83	49	7709
300099	206613	Vistara	UK	826	Chennai	Afternoon	one	Morning	Hyderabad	Economy	20.58	49	8640
300100	206614	Vistara	UK	822	Chennai	Morning	one	Morning	Hyderabad	Economy	23.33	49	8640
300101	206615	Vistara	UK	824	Chennai	Night	one	Night	Hyderabad	Economy	24.42	49	8640

300102 rows × 13 columns

- Kiểm tra và loại bỏ dữ liệu trùng lặp:

```
# check duplicates
df.duplicated().sum()
# drop duplicates
df.drop_duplicates(inplace=True)
```

- Kiểm tra giá trị null:

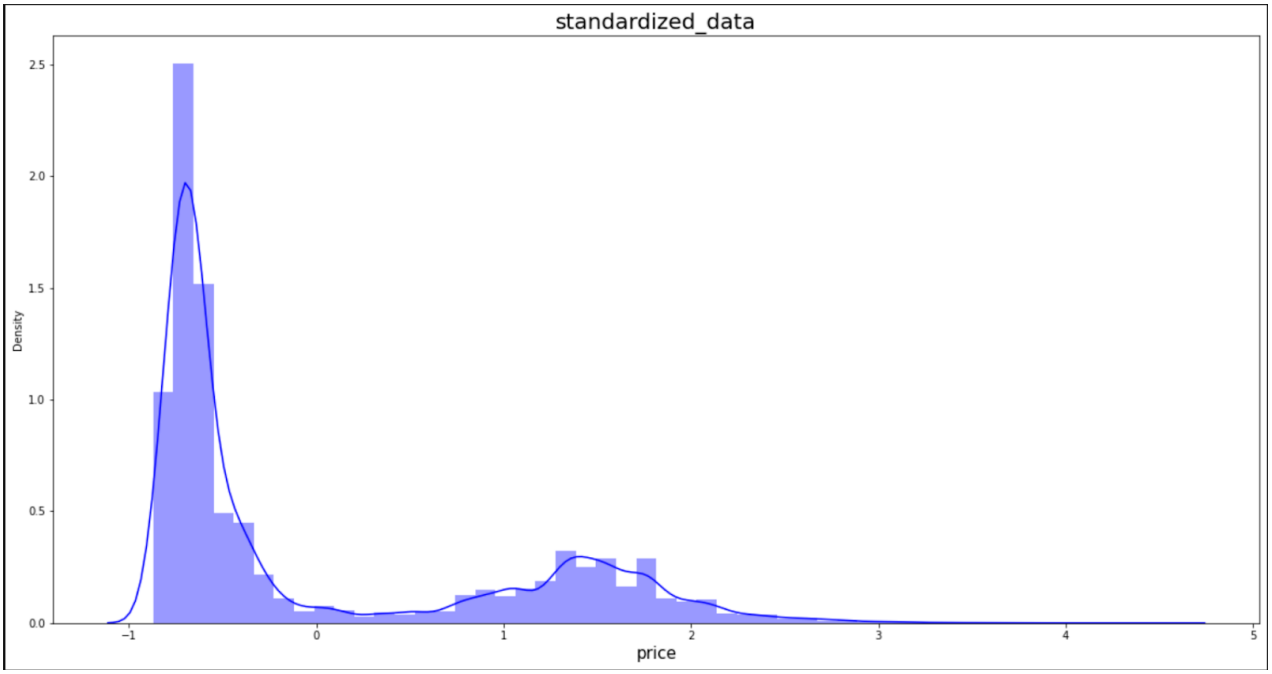
```
# Check missing values

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

```
airline             0
source_city         0
departure_time      0
stops               0
arrival_time        0
destination_city    0
class               0
duration            0
days_left           0
price               0
flight              0
dtype: int64
```

- Data scaling:

```
#standardize the data
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

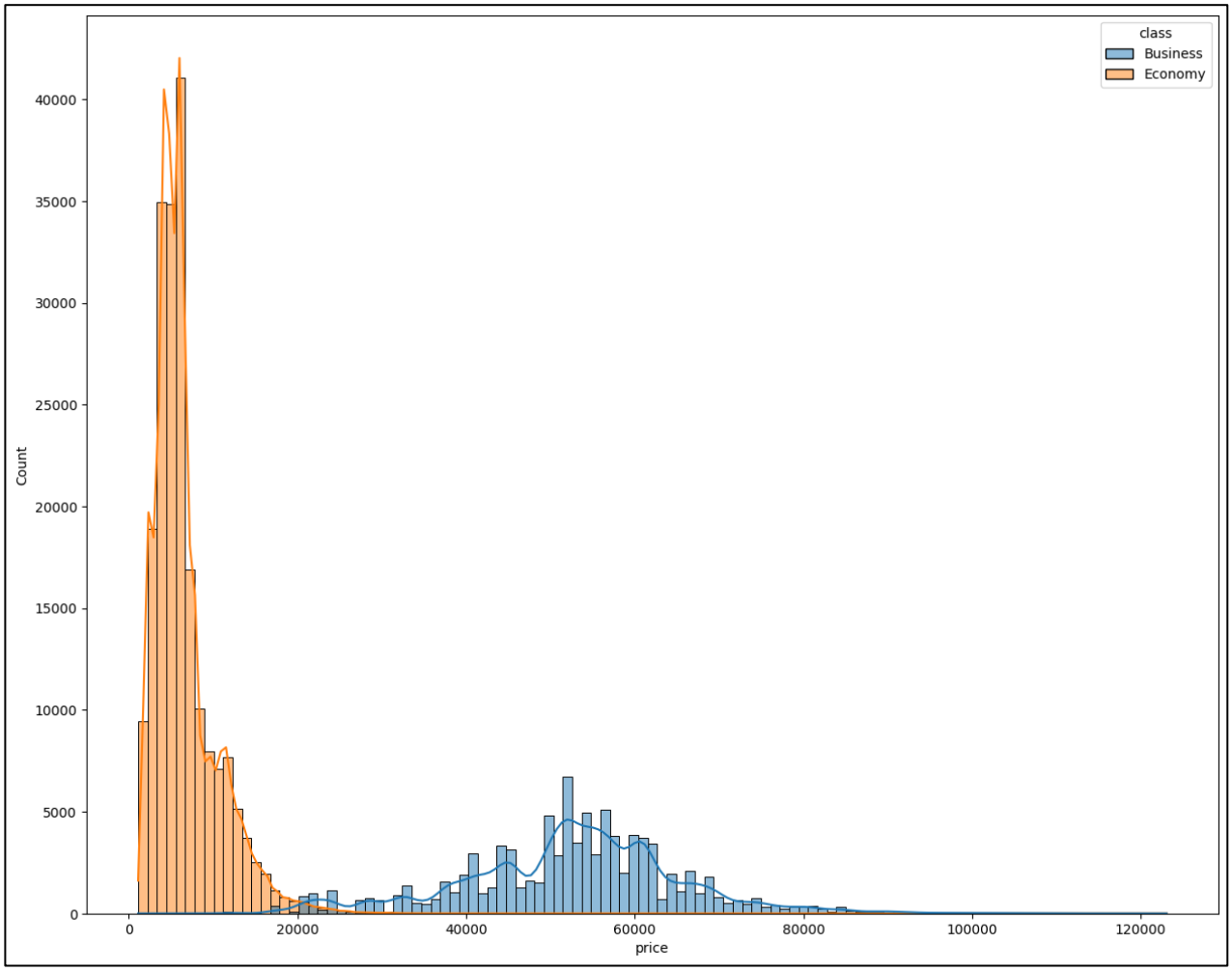
```
x_train=scaler.fit_transform('price')
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
plt.figure(figsize=(20,10))

sns.regplot(x='price',data=result,color=blue)
```



Hình 2. 2: Standardized_Data

2.5. Phân tích ban đầu:

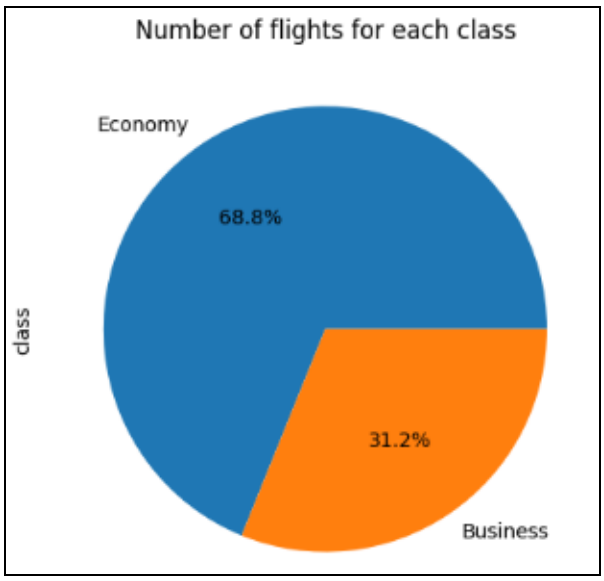
- Phân phối giá vé của 2 loại vé phổ thông và thương gia:



Hình 2. 3: Biểu đồ phân phối giá vé máy bay theo hai hạng vé

⇨ Nhìn chung giá vé thương gia cao hơn nhiều so với giá vé phổ thông, do đó mức độ phổ biến không bằng. Phân phối mức giá vé trải rộng nhiều mức giá khác nhau.

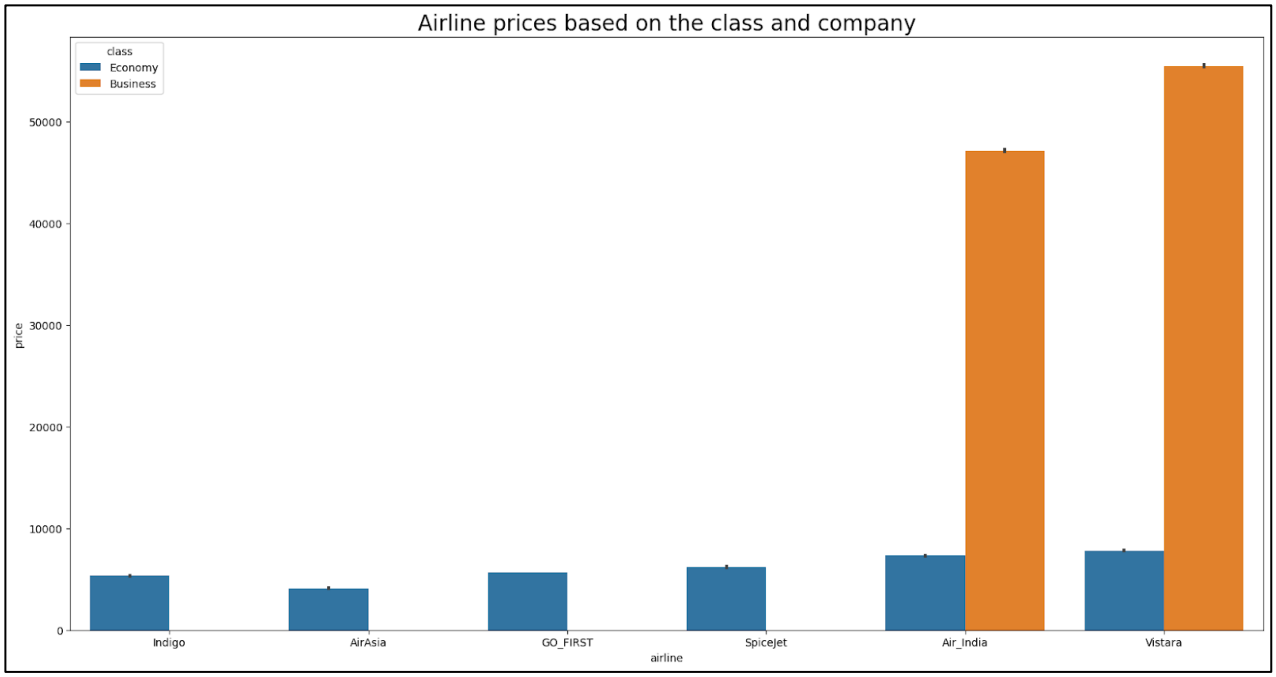
- Tỉ lệ tổng số lượng vé thuộc 2 hạng:



Hình 2. 4: Biểu đồ thể hiện số chuyến bay theo từng hạng vé

⇨ Số lượng chuyến bay có hạng vé phổ thông chiếm số lượng lớn, gấp đôi số lượng chuyến bay có hạng vé thương gia.

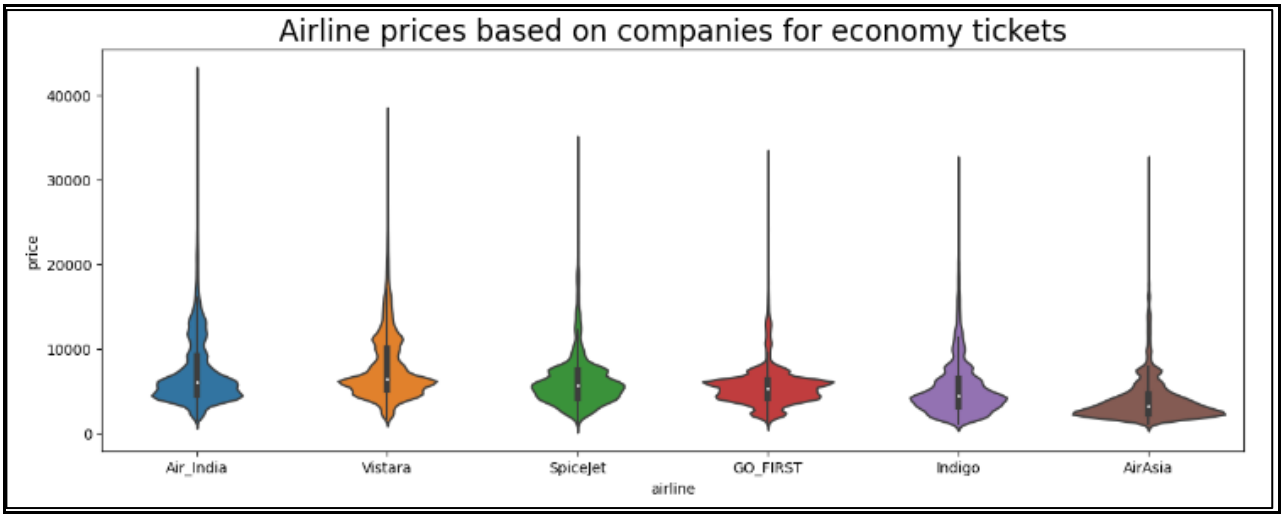
- Giá vé giữa hạng Phổ thông và hạng Thương gia khác nhau như thế nào?



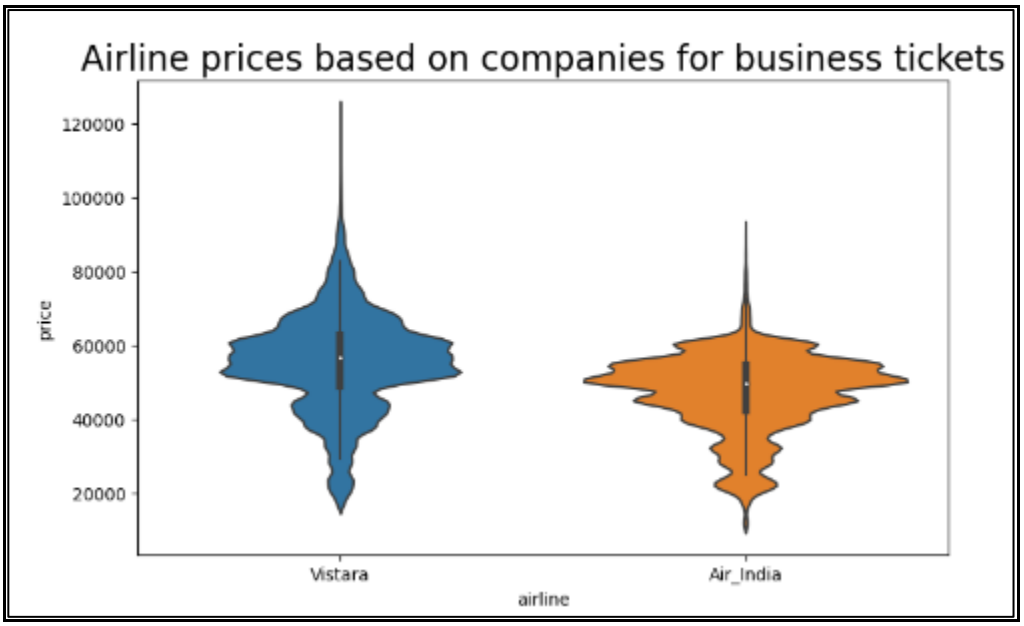
Hình 2. 5: Biểu đồ giá vé bay giữa hạng thương gia và phổ thông của từng hãng hàng không

⇨ Các chuyến bay thương gia chỉ có ở hai hãng: Air India và Vistara. Ngoài ra, có một khoảng cách lớn giữa giá ở hai hạng vé, giá vé thương gia cao gần gấp 5 lần giá vé Phổ thông.

- Giá có thay đổi với các hãng hàng không không?



Hình 2. 6: Biểu đồ thể hiện giá vé máy bay phổ thông của mỗi hãng hàng không

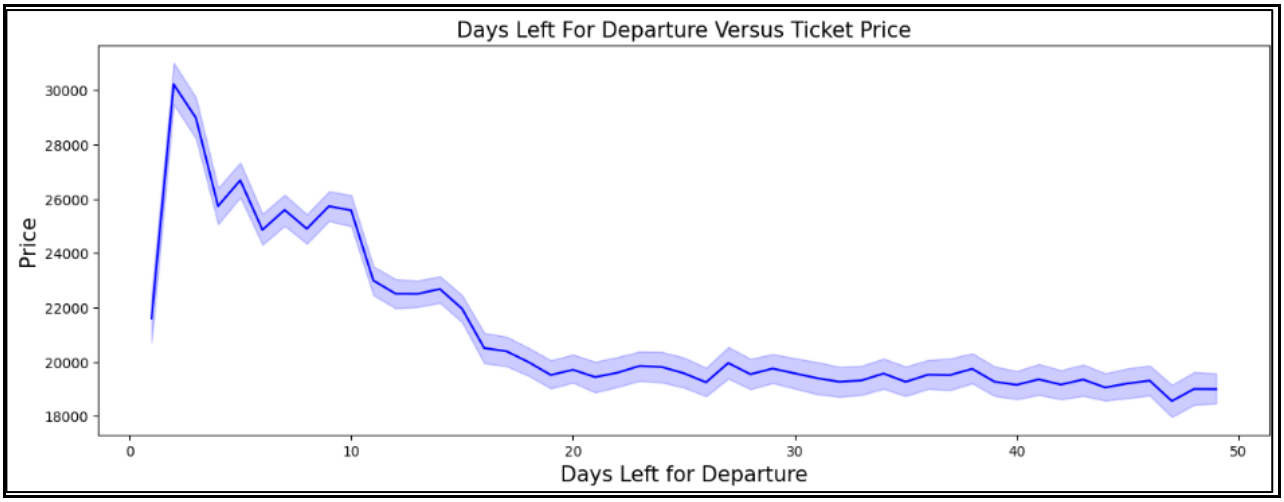


Hình 2. 7: Biểu đồ thể hiện giá vé máy bay thương gia của hai hãng hàng không

⇨ Có sự khác biệt nhỏ giữa mỗi hãng hàng không trên biểu đồ này, AirAsia dường như có các chuyến bay rẻ nhất trong khi Air India và Vistara đắt hơn. Tuy nhiên,

có vẻ như vé hạng thương gia của Vistara đắt hơn một chút so với vé của Air India.

- Giá bị ảnh hưởng như thế nào khi vé được mua chỉ trong 1 hoặc 2 ngày trước ngày khởi hành?

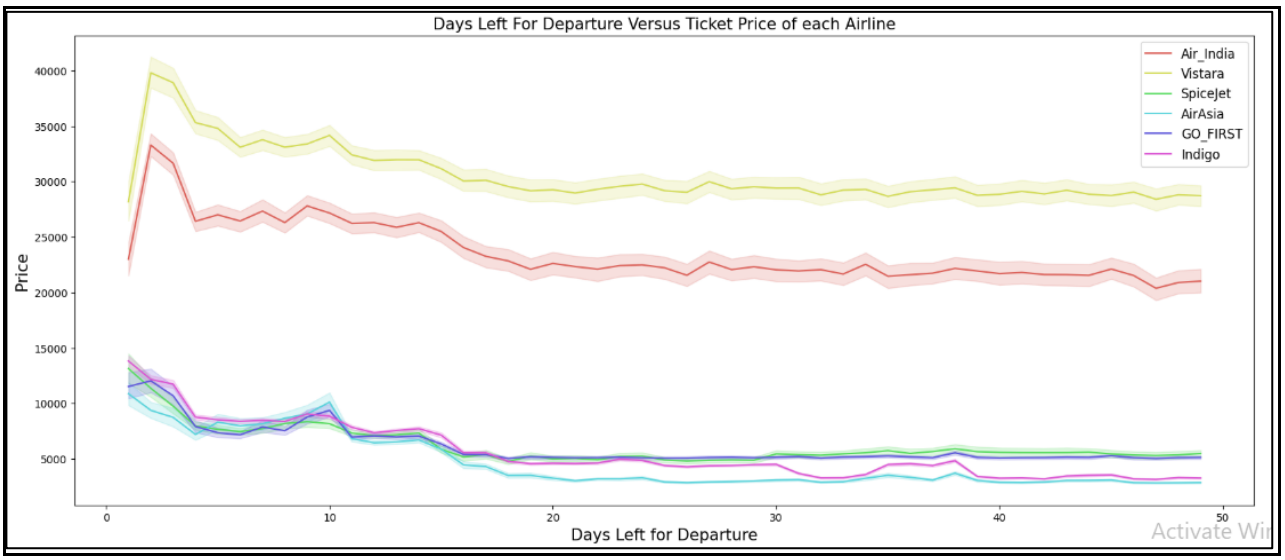


Hình 2. 8: Biểu đồ thể hiện giá vé máy bay tùy thuộc ngày còn lại so với ngày khởi hành

⇨ Một biểu đồ có thể nhìn thấy rõ sự chênh lệch trong giá vé máy bay tùy thuộc vào số ngày còn lại so với ngày khởi hành.

⇨ Biểu đồ cho thấy giá tăng chậm và ổn định trong khoảng từ 20-50 ngày đặt vé trước và sau đó bắt đầu tăng mạnh khi khoảng thời gian còn lại trước khi khởi hành giảm là 20 ngày trước chuyến bay, đạt đỉnh khi còn 2 ngày. Để có được mức giá rẻ, chúng ta nên đặt vé ít nhất 20 ngày trước cuộc hành trình.

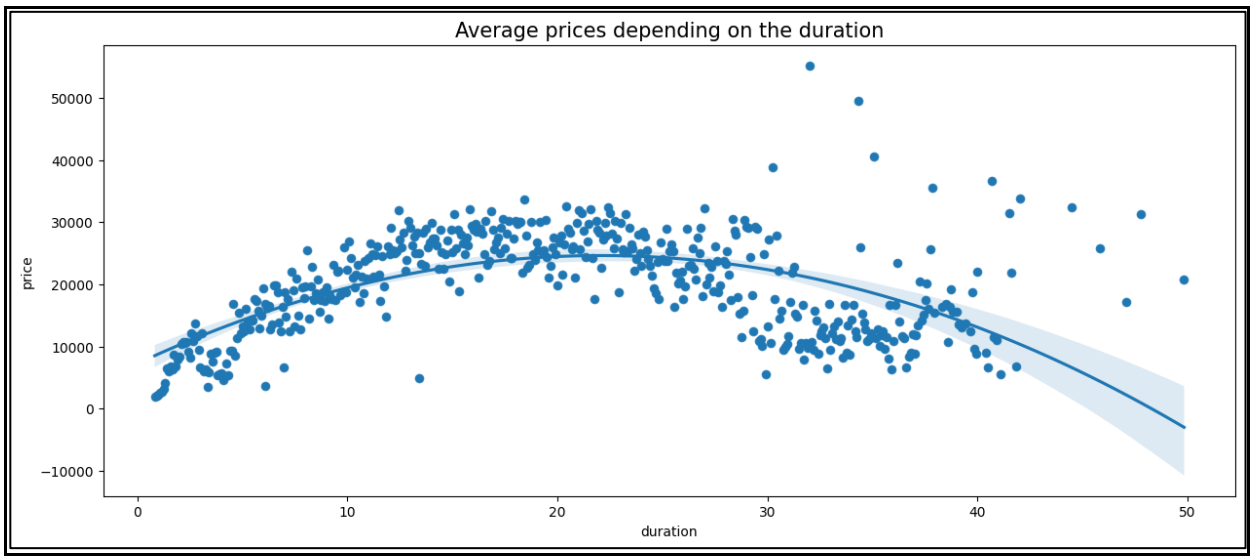
⇨ Một ngày trước chuyến bay giá vé giảm ba lần so với ngày trước đó. Điều này có thể giải thích là do các hãng muốn lấp đầy ghế trống nên hạ giá vé để đảm bảo máy bay luôn đầy khách.



Hình 2. 9: Biểu đồ thể hiện chi tiết giá vé máy bay tùy thuộc số ngày còn lại so với ngày khởi hành của từng hãng hàng không

⇨ Như chúng ta có thể thấy khi so sánh với các hãng khác khi còn hai ngày nữa là khởi hành thì giá vé rất cao đối với tất cả các hãng hàng không.

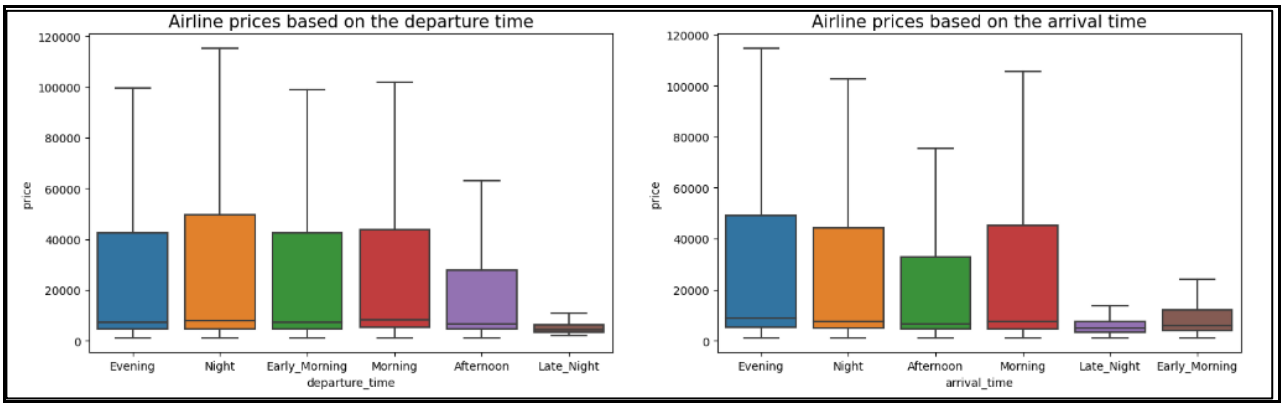
- Giá có thay đổi theo thời gian của chuyến bay không?



Hình 2. 10: Biểu đồ giá vé trung bình dựa trên thời gian bay

⇨ Rõ ràng là ở đây mối quan hệ không phải là tuyến tính mà có thể xấp xỉ bằng một đường cong bậc hai. Giá đạt mức giá cao trong khoảng thời gian 20 giờ trước khi giảm trở lại. Tuy nhiên, một số ngoại lệ dường như ảnh hưởng đến đường cong hồi quy.

- Giá vé có thay đổi theo thời gian khởi hành và thời gian hạ cánh không?



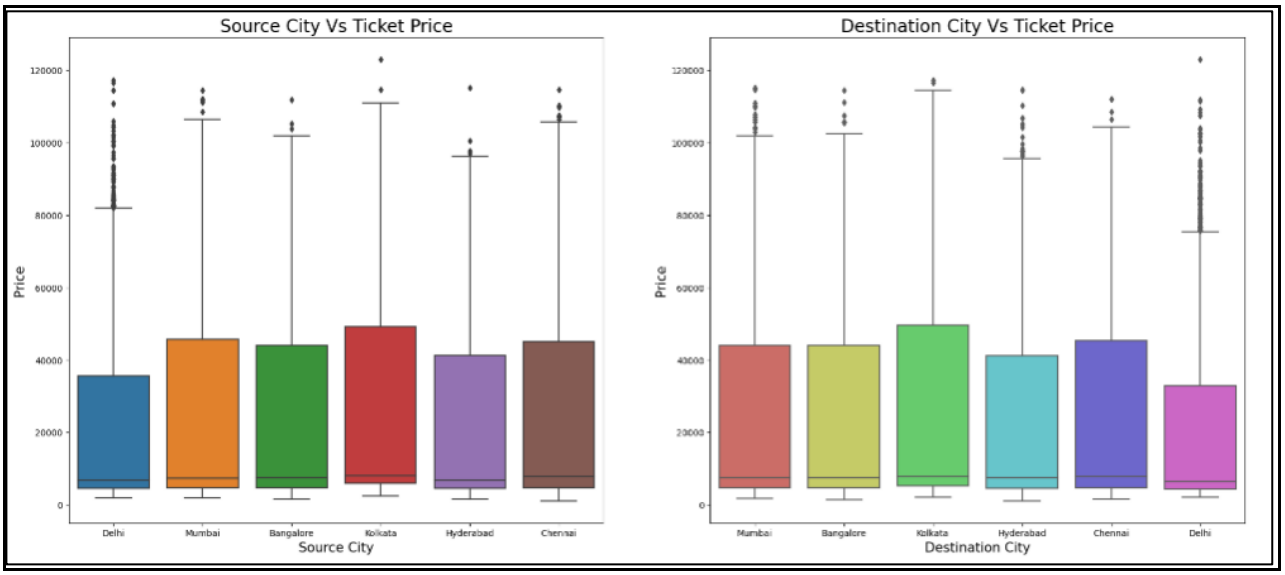
Hình 2. 11: Biểu đồ giá vé theo thời gian khởi hành và hạ cánh

⇨ Khởi hành hoặc đến đêm khuya vẫn là cách rẻ nhất để đi du lịch. Nhưng cũng có thể thấy rằng đến sáng sớm cũng khá rẻ và chuyến bay chiều rẻ hơn một chút so với buổi tối, buổi sáng và đêm.

+ Chuyến bay đêm khuya là rẻ nhất.
+ Các chuyến bay vào buổi tối là đắt nhất.
+ Giá vé cao hơn cho các chuyến bay khi thời gian khởi hành là vào ban đêm.
+ Giá vé gần như bằng nhau cho các chuyến bay có giờ khởi hành sáng sớm, sáng và tối.
+ Giá vé thấp cho các chuyến bay có giờ khởi hành đêm khuya.
+ Giá vé cao hơn cho các chuyến bay khi thời gian đến là vào buổi tối.
+ Giá vé gần như bằng nhau đối với các chuyến bay có giờ đến là sáng và tối.

+ Giá vé thấp cho các chuyến bay có giờ đến cuối đêm trùng với giờ khởi hành.

- Giá thay đổi như thế nào với sự thay đổi trong điểm xuất phát và điểm hạ cánh?



Hình 2. 12: Biểu đồ giá vé theo điểm xuất phát và điểm hạ cánh

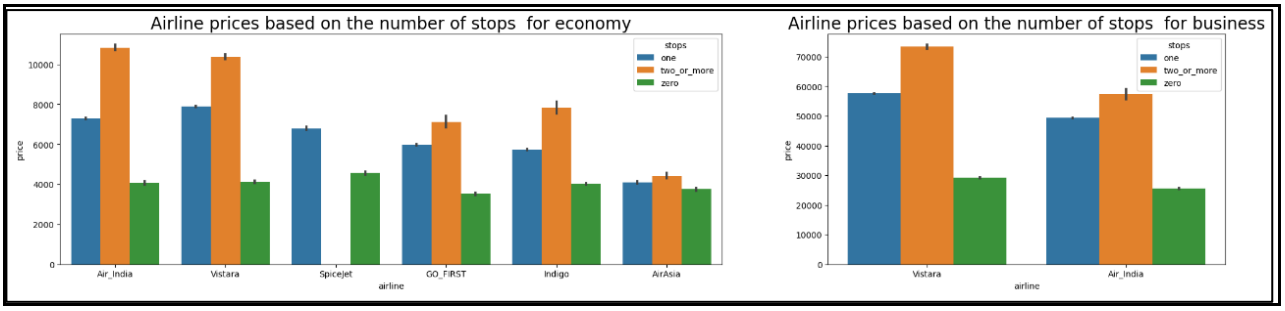
⇨ Điểm xuất phát với giá vé: Nhìn chung các chuyến bay khởi hành từ Delhi thường rẻ hơn so với các chuyến bay từ các thành phố nguồn khác và thủ đô này cũng là điểm đến rẻ nhất có lẽ vì là thủ đô, sân bay lớn nhất và có nhiều chuyến bay hơn. Mặt khác, giá cả ít nhiều giống nhau các chuyến bay bắt đầu từ Chennai đắt nhất. Giá vé cao hơn cho các chuyến bay có Thành phố đi là Kolkata. Giá vé gần như bằng nhau cho các chuyến bay Có thành phố đi như Mumbai và Chennai, Hyderabad và Bangalore.

⇨ Điểm hạ cánh với giá vé:

+ Các chuyến bay có điểm đến Delhi là rẻ nhất.
+ Các chuyến bay có điểm đến là Kolkata đắt nhất.

+ Giá vé cao hơn cho các chuyến bay có thành phố đến là Kolkata và Chennai.
+ Giá vé gần như bằng nhau cho các chuyến bay có điểm đến là Mumbai và Bangalore.
+ Giá vé thấp cho các chuyến bay có thành phố đến là Delhi.

- Số điểm dừng có ảnh hưởng đến giá không?



Hình 2. 13: Biểu đồ giá vé theo từng chặng dừng

⇨ Rõ ràng là càng nhiều điểm dừng thì chuyến bay càng đắt, ngoại trừ AirAsia nơi giá dường như không đổi. Hành vi và phân tích khác nhau của AirAsia có xu hướng cho thấy rằng nó liên quan đến một công ty chi phí thấp.

2.6. Phân tích khám phá dữ liệu:

2.6.1. Mô tả thống kê:

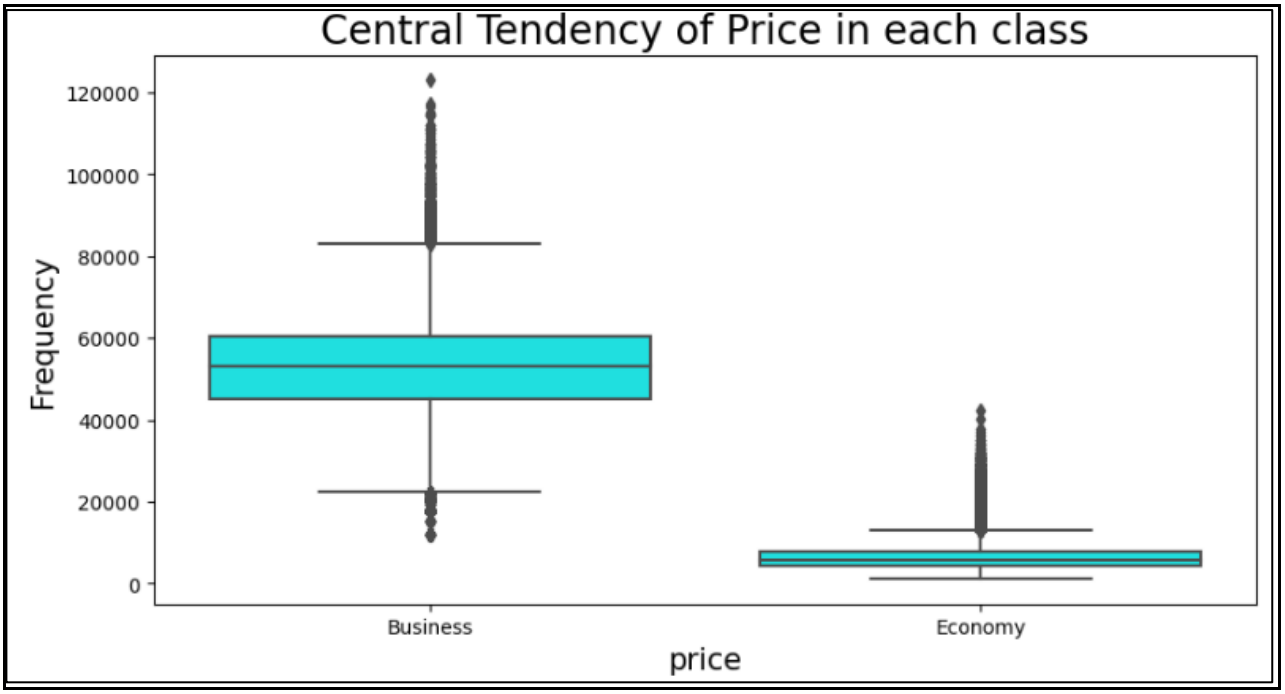
- Kiểm tra số giá trị không trùng của mỗi cột:

```
df.nunique()
```

```
airline             6
source_city         6
departure_time      6
stops               3
arrival_time        6
destination_city    6
class               2
duration          476
days_left          49
price           12157
flight           1560
dtype: int64
```

- Tìm ra xu hướng trung tâm của giá trong mỗi hạng vé:

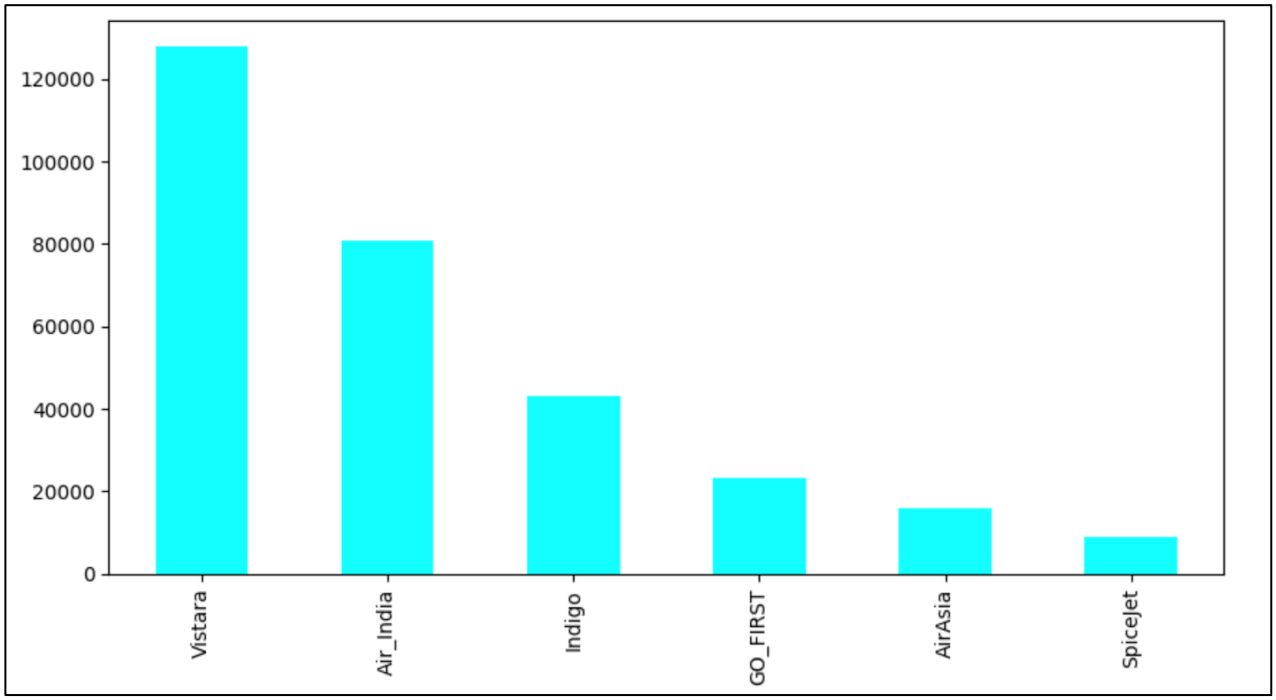
```
plt.figure(figsize=(10,5))
sns.boxplot(x='class',y='price',data=df,color='cyan')
plt.title('Central Tendency of Price in each class',fontsize=20)
plt.xlabel('price',fontsize=15)
plt.ylabel('Frequency',fontsize=15)
plt.show()
```



Hình 2. 14: Xu hướng trung tâm của giá trong mỗi hạng vé

- Trực quan hóa số lượng chuyến bay của từng hãng:

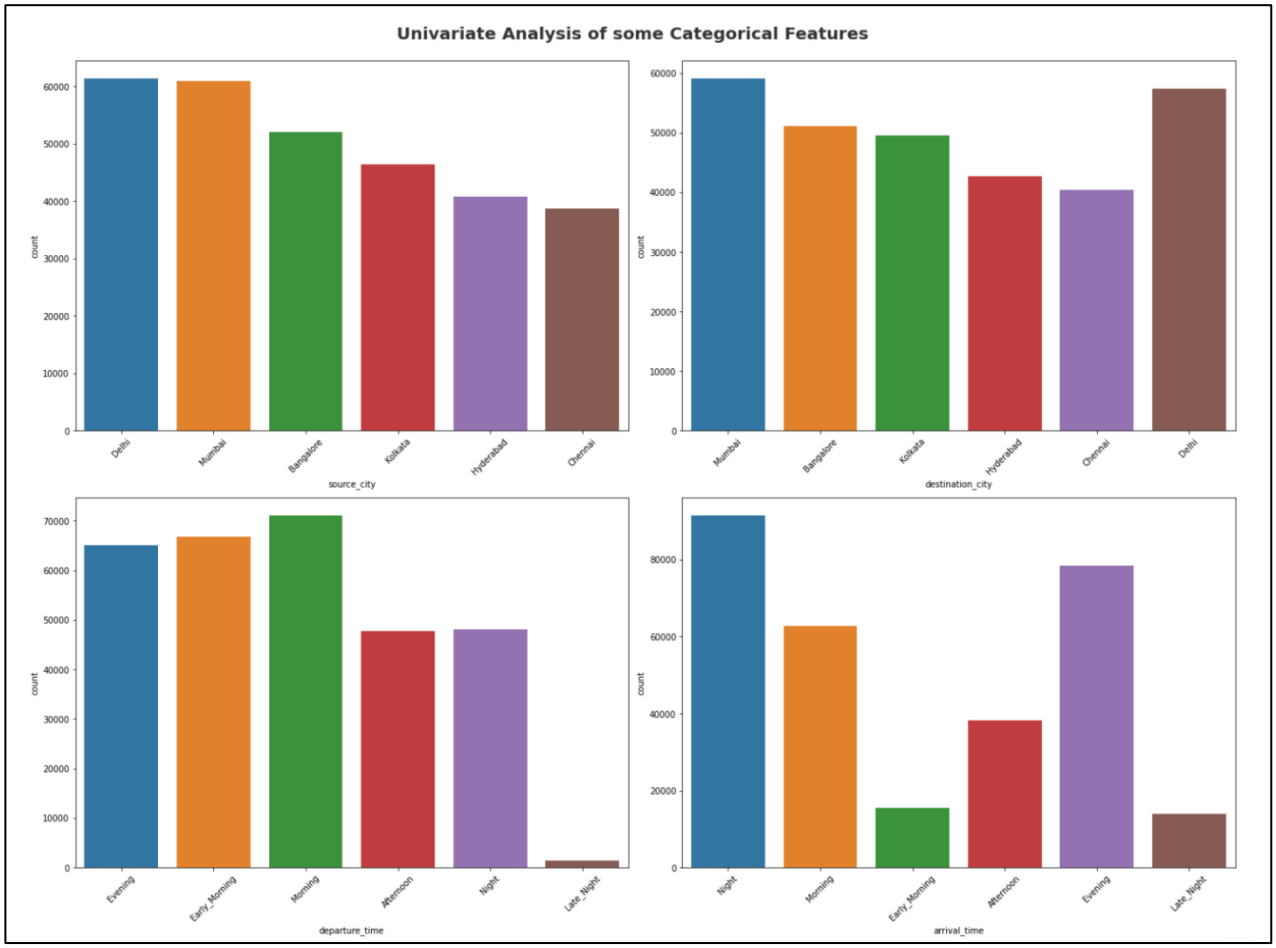
```
df['airline'].value_counts().plot(kind='bar', figsize=(10,5),
color='cyan')
```



Hình 2. 15: Số lượng chuyến bay của từng hãng

- Phân tích biến đơn một số Features phân loại:

```
plt.figure(figsize=(20,15))
plt.suptitle('Univariate Analysis of some Categorical Features',
fontsize=20, fontweight='bold', alpha=0.8, y=1.)
cat1 = ['source_city', 'destination_city', 'departure_time',
'arrival_time']
for i in range(0, len(cat1)):
   plt.subplot(2,2,i+1)
   sns.countplot(x=df[cat1[i]])
   plt.xlabel(cat1[i])
   plt.xticks(rotation=45)

   plt.tight_layout()
```



Hình 2. 16: Phân tích biến đơn một số Features phân loại

⇨ Từ hình ảnh đã được trực quan hóa, ta nhận thấy:

+ Với [Source_city], số lượng khá đều nhau giữa các thành phố, Delhi và Mumbai có số lượng lớn nhất.

+ Với [Destination_city], số lượng phân bố khá đều nhau giữa các thành phố, Delhi và Mumbai có số lượng lớn nhất.

+ Với [Departure_time], thời gian bắt đầu bay tương đối đều giữa các thời gian trong ngày. Buổi sáng (Morning) và sáng sớm (Early_morning) có tổng số các chuyến bay cao nhất, số chuyến bay đêm khuya (Late_Night) rất ít so với tổng số quan sát.

+ Với [Arrival_time], thời gian hạ cánh vào chiều tối [Evening] và ban đêm [Night] chiếm số lượng lớn các chuyến bay, đêm khuya [Late_Night] là thời điểm máy bay ít hạ cánh nhất.

2.6.2. Feature engineering:

- Thông tin các thuộc tính:

  + Airline: Hãng máy bay (có tất cả 6 hãng)
  + Source_city: thành phố khởi hành (6 thành phố)
  + Departure_time: Thời gian cất cánh (chia thành 6 khung giờ)
  + Stops: số trạm dừng (3 giá trị: zero, one và two_or_more)
  + Destination_city: điểm đến (6 thành phố)
  + Class: hạng vé (Business hoặc là Economy)
  + Duration: thời gian bay
  + Days_left: số ngày cho đến ngày bay (được tính bằng ngày sẽ bay trừ đi ngày đặt vé)
  + Price: giá vé
  + Flight: mã chuyến bay

- Bỏ cột “Flight" (Mã chuyến bay) không cần thiết

```
df.drop(['flight'], axis=1, inplace=True)
```

- Tứ phân vị của 3 biến duration, days_left, price:

```
df.describe()
```

	duration	days_left	price
count	300102.000000	300102.000000	300102.000000
mean	12.222089	26.003859	20892.497374
std	7.192126	13.561643	22698.647167
min	0.830000	1.000000	1105.000000
25%	6.830000	15.000000	4783.000000
50%	11.250000	26.000000	7425.000000
75%	16.170000	38.000000	42521.000000
max	49.830000	49.000000	123071.000000

- Xác định dữ liệu số và dữ liệu phân loại:

```
numeric_features=[feature for feature in df.columns if
df[feature].dtype != 'O']
categorical_features=[feature for feature in df.columns if
df[feature].dtype == 'O']
print(f'We have {len(numeric_features)} numerical features
:{numeric_features}')

print(f'We  have  {len(categorical_features)}  categorical  features
:{categorical_features}')
```

```
We have 3 numerical features :['duration', 'days_left', 'price']
We have 7 categorical features :['airline', 'source_city', 'departure_time', 'stops', 'arrival_time', 'destination_city', 'class']
```

- Các biến phân loại: Các biến phân loại trong tập dữ liệu: 'airline', 'source_city', 'departure_time', 'stops', 'arrival_time', 'destination_city', 'class'.

    + Sử dụng phương pháp One-hot Encoding để mã hóa và đưa dữ liệu phân loại về dạng số ('departure_time', 'stops', 'arrival_time', 'class').

    + Sử dụng phương pháp mã hóa bằng gán nhãn (LabelEncoder) với các biến: 'departure_time', 'stops', 'arrival_time', 'class'

- LabelEncoder: Đếm số giá trị duy nhất trong cột “stops"

```
# unique values in stops column

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

```
one            250812
zero            36004
two_or_more     13286
Name: stops, dtype: int64
```

Thực hiện mã hóa bằng gán nhãn (Label encoding) với cột “stops"

```
# Label Encoding with Stops
df.replace({'stops': {'zero': 0, 'one': 1, 'two_or_more':
2}}, inplace=True)
```

- Làm tương tự với các thuộc tính còn lại thu được bảng dữ liệu sau:

	airline	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
0	Air_India	Delhi	4	0	4	Mumbai	1	2.00	1	25612
1	Air_India	Delhi	4	0	5	Mumbai	1	2.25	1	25612
2	Air_India	Delhi	4	1	5	Mumbai	1	24.75	1	42220
3	Air_India	Delhi	5	1	5	Mumbai	1	26.50	1	44450
4	Air_India	Delhi	4	1	5	Mumbai	1	6.67	1	46690
...	...	...	...	...	...	...	...	...	...	...
300097	Vistara	Chennai	1	1	5	Hyderabad	0	13.83	49	7697
300098	Vistara	Chennai	1	1	5	Hyderabad	0	13.83	49	7709
300099	Vistara	Chennai	3	1	2	Hyderabad	0	20.58	49	8640
300100	Vistara	Chennai	2	1	2	Hyderabad	0	23.33	49	8640
300101	Vistara	Chennai	5	1	5	Hyderabad	0	24.42	49	8640

300102 rows × 10 columns

- One-hot Encoding với các cột ‘airline’, ‘source_city’, ‘destination_city’:
  + Vẽ biểu đồ tương quan:

```
df_corr = df.corr()
df_corr.style.background_gradient()
```

	departure_time	stops	arrival_time	class	duration	days_left	price
departure_time	1.000000	-0.069016	-0.079934	0.031051	0.132908	-0.000274	0.021050
stops	-0.069016	1.000000	0.046393	0.001049	0.468120	-0.008553	0.119678
arrival_time	-0.079934	0.046393	1.000000	-0.022285	-0.123735	-0.000812	-0.000807
class	0.031051	0.001049	-0.022285	1.000000	0.138625	-0.012995	0.937856
duration	0.132908	0.468120	-0.123735	0.138625	1.000000	-0.039103	0.204136
days_left	-0.000274	-0.008553	-0.000812	-0.012995	-0.039103	1.000000	-0.091906
price	0.021050	0.119678	-0.000807	0.937856	0.204136	-0.091906	1.000000

⇨ Yếu tố Class có ảnh hưởng lớn đến giá vé máy bay, tiếp đó là thời gian bay.

+ One-hot:

```
# One hot encoding

df = pd.get_dummies(df, columns=['airline', 'source_city',
'destination_city'], drop_first=True)
df
```

	departure_time	stops	arrival_time	class	duration	days_left	price	airline_Air_India	airline_GO_FIRST	airline_Indigo	...	source_city_Chennai	source_city_Delhi	source_city_Hyderabad
0	4	0	4	1	2.00	1	25612	1	0	0	...	0	1	0
1	4	0	5	1	2.25	1	25612	1	0	0	...	0	1	0
2	4	1	5	1	24.75	1	42220	1	0	0	...	0	1	0
3	5	1	5	1	26.50	1	44450	1	0	0	...	0	1	0
4	4	1	5	1	6.67	1	46690	1	0	0	...	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
300097	1	1	5	0	13.83	49	7697	0	0	0	...	1	0	0
300098	1	1	5	0	13.83	49	7709	0	0	0	...	1	0	0
300099	3	1	2	0	20.58	49	8640	0	0	0	...	1	0	0
300100	2	1	2	0	23.33	49	8640	0	0	0	...	1	0	0
300101	5	1	5	0	24.42	49	8640	0	0	0	...	1	0	0

300102 rows × 22 columns

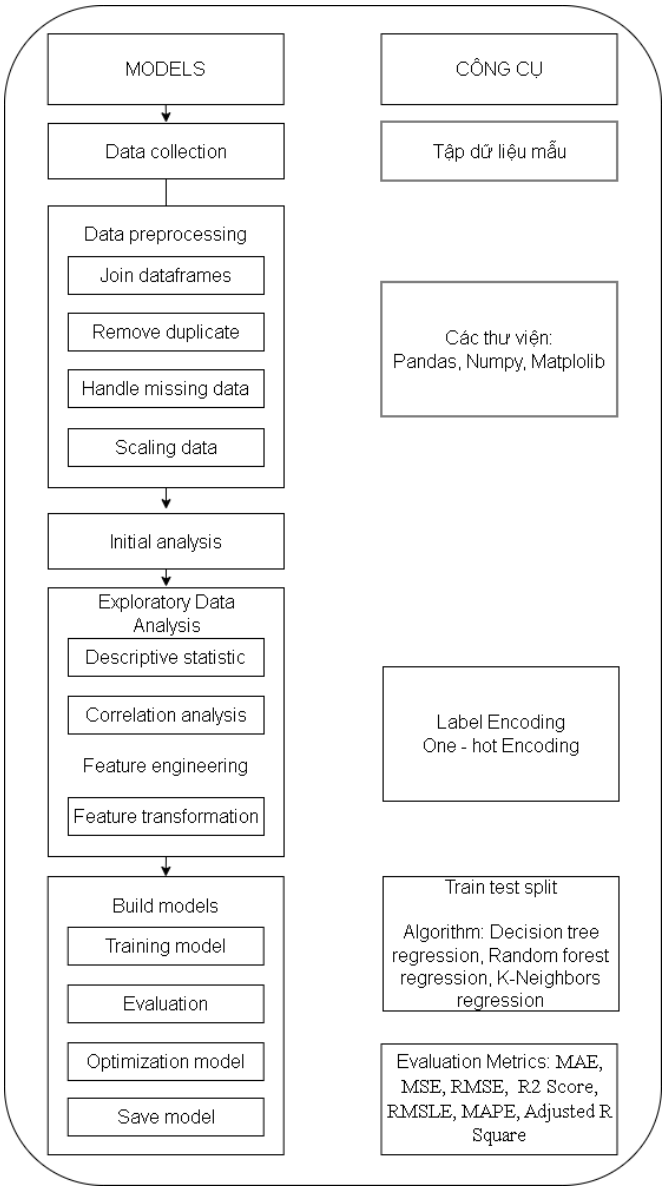
source_city_Hyderabad	source_city_Kolkata	source_city_Mumbai	destination_city_Chennai	destination_city_Delhi	destination_city_Hyderabad	destination_city_Kolkata	destination_city_Mumbai
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0

CHƯƠNG 3: THỰC NGHIỆM VÀ ĐÁNH GIÁ MÔ HÌNH

Tóm tắt chương 3

Tiến hành thực nghiệm dữ liệu và cung cấp các biểu đồ trực quan hóa. Triển khai các thuật toán Decision Tree, Random Forest, K-Nearest Neighbors và đưa ra các mô hình cụ thể. Từ đó, so sánh và lựa chọn các mô hình phù hợp với tập dữ liệu và đánh giá kết quả mô hình.

3.1. Quy trình thực nghiệm:



Hình 3. 1: Mô tả quy trình thực nghiệm

3.2. Xây dựng mô hình dự báo:

- Biến phụ thuộc là giá vé ‘price’ và các biến độc lập là các cột còn lại trong tập dữ liệu:

```
x = df.drop(['price'], axis=1)

y = df['price']
```

- Phân chia tập dữ liệu thành tập Train và test với thư viện Train_test_split. Tỷ lệ tập train và test là 80:20

```
# import libaries

from sklearn.model_selection import train_test_split
# train test split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42)
```

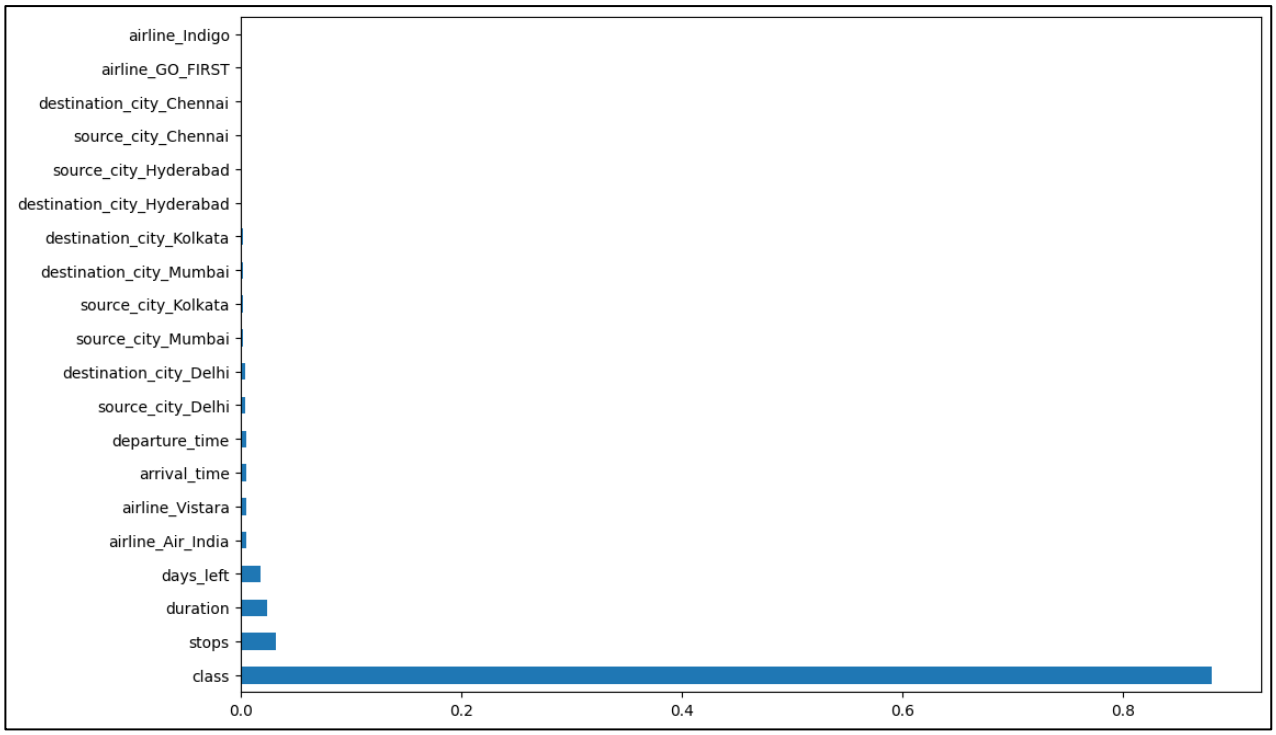
- Feature importance: hỗ trợ xác định mức độ quan trọng của các features, từ đó, có thể cải thiện độ chính xác của mô hình và ứng dụng vào trong phân tích dữ liệu sau này. (thư viện sử dụng Extra Tree Regression)

```
# import libaries
from sklearn.ensemble import ExtraTreesRegressor
selection = ExtraTreesRegressor()

selection.fit(x, y)
```

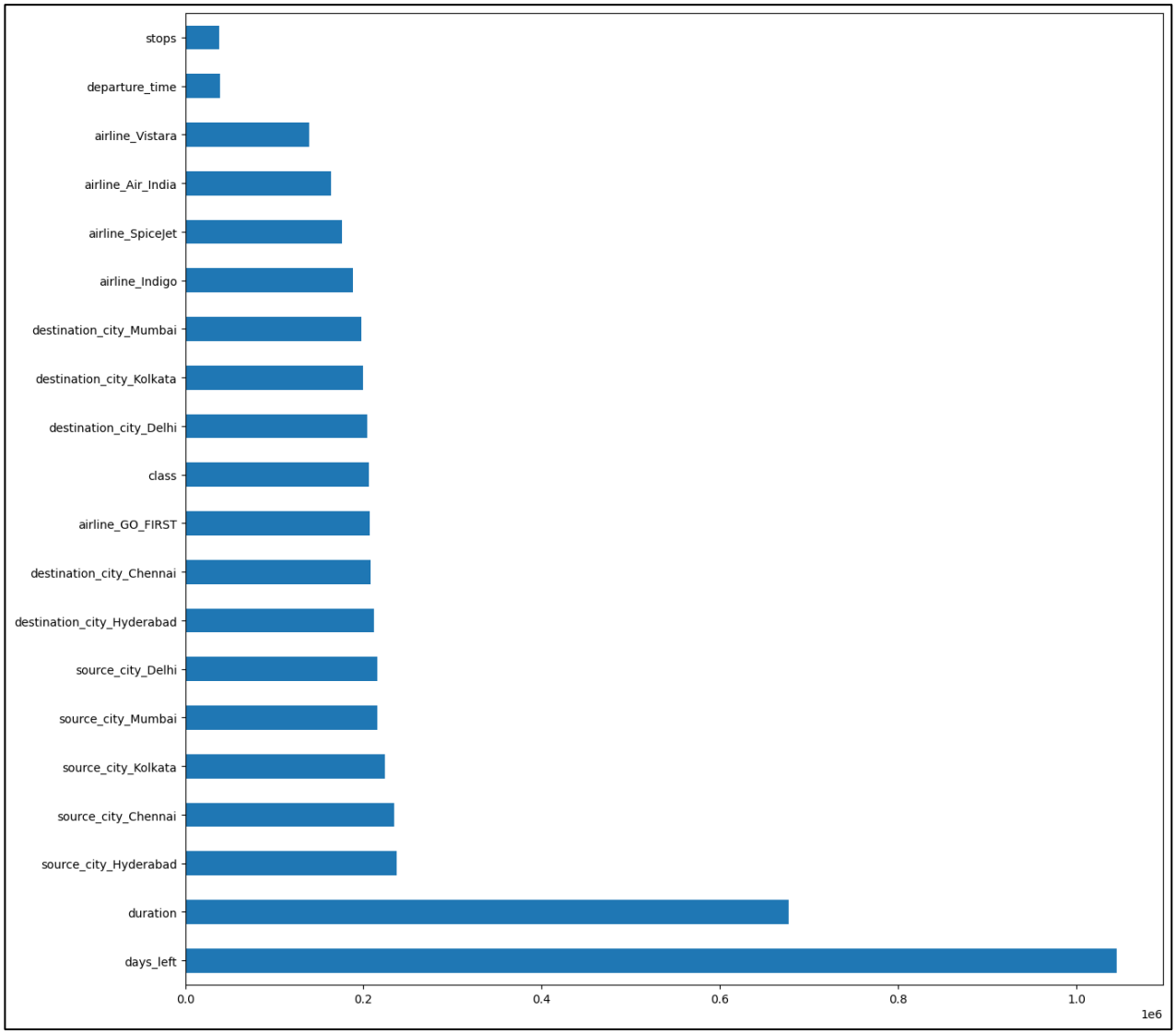
```
▾ ExtraTreesRegressor
ExtraTreesRegressor()
```

```
#plot graph of feature importances for better visualization
    plt.figure(figsize = (12,8))
```

```
        feat_importances =
        pd.Series(selection.feature_importances_,
        index=x.columns)
        feat_importances.nlargest(20).plot(kind='barh')
plt.show()
```




```
        from sklearn.feature_selection import SelectKBest
        from sklearn.feature_selection import chi2
        #Deifne feature selection
        fs=SelectKBest(score_func=chi2)
        # Applying feature selection
        X_selected=fs.fit(x,y)

        plt.figure(figsize=(15,15))
        feat_importances = pd.Series(X_selected.scores_,
           index=x.columns)
        feat_importances.nlargest(20).plot(kind='barh')

plt.show()
```



⇨ Như đã thực nghiệm, nhóm nhận thấy các features có ảnh hưởng quan trọng hơn hẳn là 'days_left', 'duration', 'class' và 'stops'.

3.2.1. Lựa chọn mô hình:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KneighborsRegressor
```

⇨ Nhóm thực hiện chạy ba mô hình lần lượt là: DecisionTreeRegressor, RandomForestRegressor, KNeighborsRegressor

3.2.2. Thực nghiệm mô hình:

Thực nghiệm lần lượt 3 mô hình và đánh giá qua các chỉ số MAE, MSE, RMSE, R2 Score, RMSLE, MAPE, Adjusted R Square.

```
DRmodel = DecisionTreeRegressor()
RDmodel = RandomForestRegressor()
KNmodel = KNeighborsRegressor()

a={'Model Name':[], 'Mean_Absolute_Error_MAE':[]
,'Adj_R_Square':[] ,'Root_Mean_Squared_Error_RMSE':[]
,'Mean_Absolute_Percentage_Error_MAPE':[]
,'Mean_Squared_Error_MSE':[]
,'Root_Mean_Squared_Log_Error_RMSLE':[] ,'R2_score':[]}
Results=pd.DataFrame(a)
Results.head()
```


```
MD = [DRmodel, RDmodel, KNmodel]

for models in MD:
# Fit the model with train data
models.fit(x_train, y_train)
# Predict the model with test data
```

```
      y_pred = models.predict(x_test)
      # Print the model name
      print('Model Name: ', models)
      # Evaluation metrics for Regression analysis
      from sklearn import metrics

      print('Mean Absolute Error (MAE):',
round(metrics.mean_absolute_error(y_test, y_pred),3))
    print('Mean Squared Error (MSE):',
round(metrics.mean_squared_error(y_test, y_pred),3))
    print('Root Mean Squared Error (RMSE):',
round(np.sqrt(metrics.mean_squared_error(y_test, y_pred)),3))
    print('R2_score:', round(metrics.r2_score(y_test, y_pred),6))
    print('Root Mean Squared Log Error (RMSLE):',
round(np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))),3))

    # Define the function to calculate the MAPE - Mean Absolute
Percentage Error
    def MAPE (y_test, y_pred):
        y_test, y_pred = np.array(y_test), np.array(y_pred)
        return np.mean(np.abs((y_test - y_pred) / y_test)) * 100
    # Evaluation of MAPE
    result = MAPE(y_test, y_pred)
    print('Mean Absolute Percentage Error (MAPE):', round(result, 2),
'%')
    # Calculate Adjusted R squared values
    r_squared = round(metrics.r2_score(y_test, y_pred),6)
    adjusted_r_squared = round(1 - (1-r_squared)*(len(y)-1)/(len(y)-
x.shape[1]-1),6)
    print('Adj R Square: ', adjusted_r_squared)
    print('-----------------------------------------------------------
-----------------------------------------')
    #-------------------------------------------------------------------
-----------------------
    new_row = {'Model Name' : models,
```

```
                'Mean_Absolute_Error_MAE' :
metrics.mean_absolute_error(y_test, y_pred),
                'Adj_R_Square' : adjusted_r_squared,
                'Root_Mean_Squared_Error_RMSE' :
np.sqrt(metrics.mean_squared_error(y_test, y_pred)),
                'Mean_Absolute_Percentage_Error_MAPE' : result,
                'Mean_Squared_Error_MSE' :
metrics.mean_squared_error(y_test, y_pred),
                'Root_Mean_Squared_Log_Error_RMSLE':
np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))),
                'R2_score' : metrics.r2_score(y_test, y_pred)}
    Results = Results.append(new_row, ignore_index=True)

    #--------------------------------------------------------------
```

3.2.3. Kết quả thực nghiệm:

- DecisionTreeRegressor:

```
Model Name:  DecisionTreeRegressor()

Mean Absolute Error (MAE): 1188.504

Mean Squared Error (MSE): 12660292.163

Root Mean Squared Error (RMSE): 3558.13

R2_score: 0.975422

Root Mean Squared Log Error (RMSLE): 8.177

Mean Absolute Percentage Error (MAPE): 7.55 %

Adj R Square:  0.97542
```

- RandomForestRegressor:

```
Model Name:  RandomForestRegressor()
```

```
Mean Absolute Error (MAE): 1086.091

Mean Squared Error (MSE): 7643018.263

Root Mean Squared Error (RMSE): 2764.601

R2_score: 0.985162

Root Mean Squared Log Error (RMSLE): 7.925

Mean Absolute Percentage Error (MAPE): 7.1 %

Adj R Square:  0.985161
```

- KneighborsRegressor:

```
Model Name:  KNeighborsRegressor()

Mean Absolute Error (MAE): 7294.157

Mean Squared Error (MSE): 113860133.214

Root Mean Squared Error (RMSE): 10670.526

R2_score: 0.77896

Root Mean Squared Log Error (RMSLE): 9.275

Mean Absolute Percentage Error (MAPE): 67.49 %

Adj R Square:  0.778945
```

3.3. So sánh và lựa chọn mô hình:

Model	MAE	MSE	RMSE	R2 Score	RMLSE	MAPE	Adjusted R Square

Decision Tree regression	1188.504	12660292.163	3558.13	0.975422	8.177	7.55 %	0.97542
Random forest regression	1086.091	7643018.263	2764.601	0.985162	7.925	7.1 %	0.985161
K-Neighbors Regression	7294.157	113860133.214	10670.526	0.77896	9.275	67.49 %	0.778945

Bảng 3. 1: Bảng so sánh các chỉ số đánh giá trong các mô hình

⇨ So sánh các chỉ số, nhóm nhận thấy Random Forest Regression có những kết quả tốt hơn.

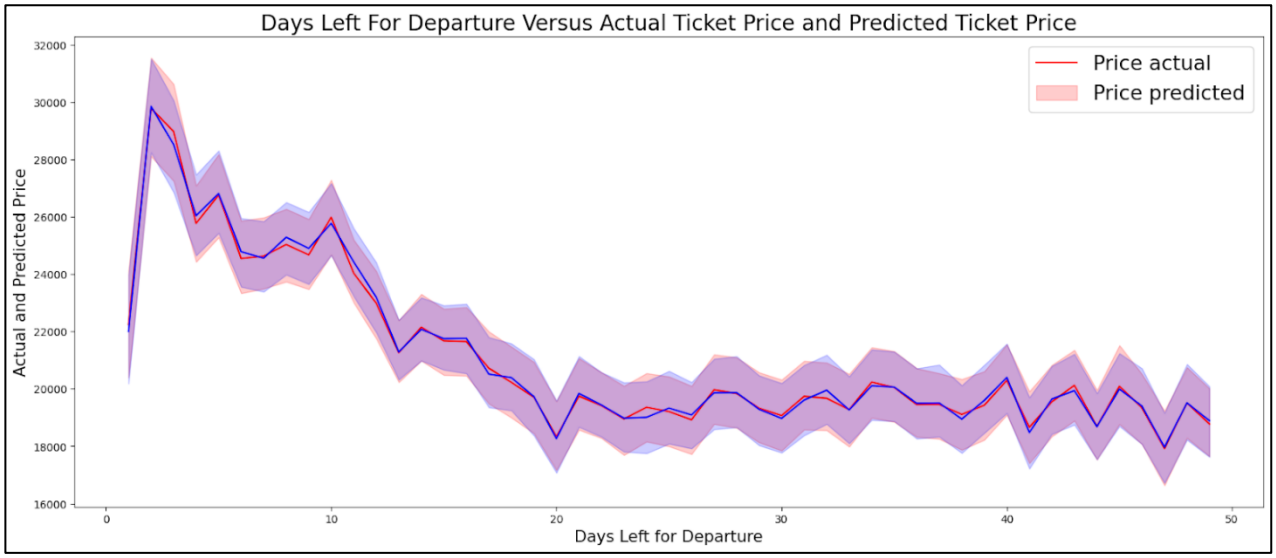
```
      #Trainig the model with
      RDmodel.fit(x_train, y_train)

      # Predict the model with test data

      y_pred = RDmodel.predict(x_test)
      out=pd.DataFrame({'Price_actual':y_test,'Price_pred':y_pred})
      result=df.merge(out,left_index=True,right_index=True)
result.sample(10)
```

	departure_time	stops	arrival_time	class	duration	days_left	price	airline_Air_India	airline_GO_FIRST	airline_Indigo	...	source_city_Hyderabad	source
227690	1	0	1	0	2.83	42	4499	0	1	0	...	0	
192955	3	1	4	0	6.67	32	2203	0	0	0	...	0	
12374	1	2	4	1	12.83	48	72339	0	0	0	...	0	
111412	2	1	4	0	6.08	39	5148	1	0	0	...	0	
2880	3	1	5	1	7.92	28	36577	0	0	0	...	0	
36757	2	0	2	1	2.08	29	23898	0	0	0	...	0	
217311	2	1	4	0	6.50	28	6461	0	0	0	...	0	
238633	5	1	3	0	19.50	25	7126	0	0	0	...	0	
147184	4	1	2	0	13.50	7	8148	0	0	0	...	0	
268840	6	0	1	0	2.17	42	4558	0	0	1	...	1	

10 rows × 24 columns

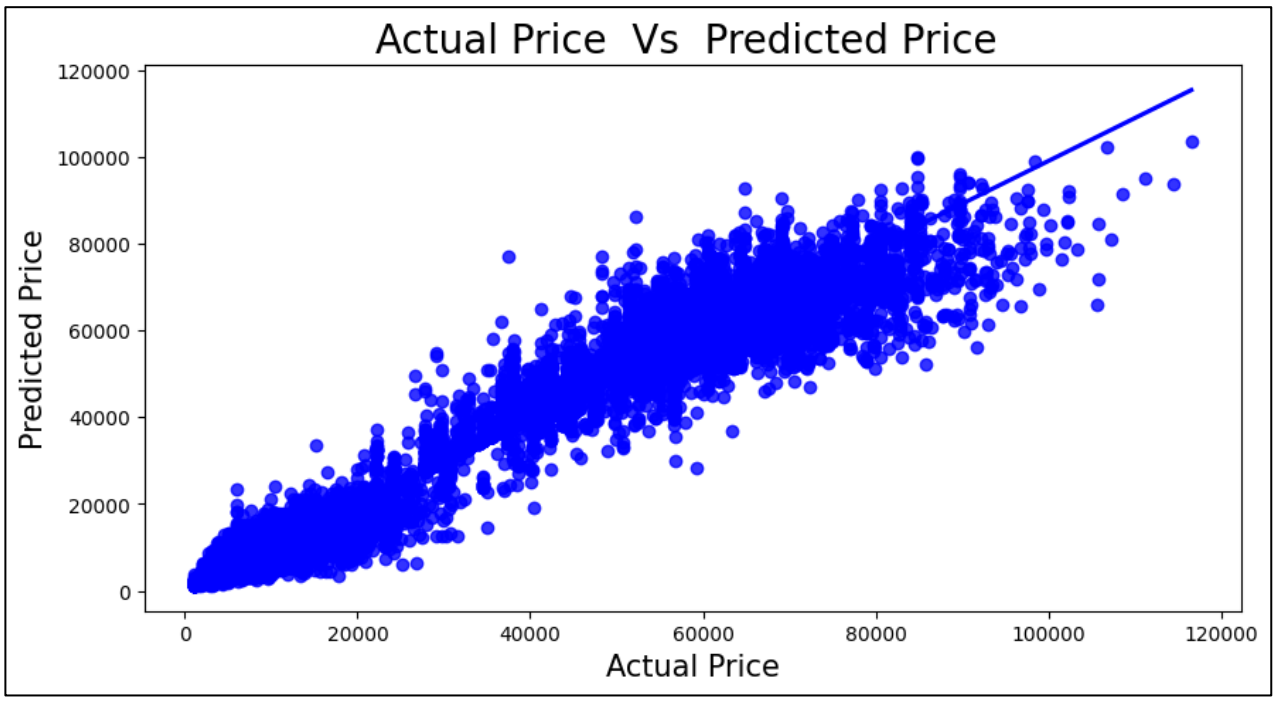
```
      plt.figure(figsize=(20,8))
```

```
        sns.lineplot(data=result,x='days_left',y='Price_actual',color='red'
        )
        sns.lineplot(data=result,x='days_left',y='Price_pred',color='blue')
        plt.title('Days Left For Departure Versus Actual Ticket Price and
        Predicted Ticket Price',fontsize=20)
        plt.legend(labels=['Price actual','Price predicted'],fontsize=19)
        plt.xlabel('Days Left for Departure',fontsize=15)
        plt.ylabel('Actual and Predicted Price',fontsize=15)
plt.show()
```



⇨ Nhìn chung kết quả dự báo tương đối chính xác, độ chênh lệch giữa giá vé máy bay dự báo và giá vé thực tế không chênh lệch nhiều.

```
        plt.figure(figsize=(10,5))
        sns.regplot(x='Price_actual',y='Price_pred',data=result,color='blue
        ')
        plt.title('Actual Price  Vs  Predicted Price ',fontsize=20)
        plt.xlabel('Actual Price',fontsize=15)
        plt.ylabel('Predicted Price',fontsize=15)
plt.show()
```



⇨ Hầu hết dữ liệu phù hợp với quy định đường tuyến tính nhưng có thể do một số ngoại lệ, một số dự báo bị lệch xa đường tuyến tính.

CHƯƠNG 4: KẾT LUẬN VÀ PHƯƠNG HƯỚNG PHÁT TRIỂN

Tóm tắt chương 4

Trong phần nội dung chương 4 này, nhóm sẽ tiến hành tóm tắt các nội dung và kết quả thực hiện được, đánh giá kết quả thực hiện, đưa ra các hạn chế còn tồn đọng và hướng phát triển của dự án.

4.1. Kết quả đạt được:

- Xác định được các yếu tố ảnh hưởng đến giá vé máy bay.
- Đưa ra được một số insight từ dữ liệu để giúp ích hơn.
- Xây dựng được mô hình dự báo vé máy bay đạt các chỉ số đánh giá như sau:

Model	MAE	MSE	RMSE	R2 Score	RMLSE	MAPE	Adjusted R Square
Random forest regression	1086.091	7643018.263	2764.601	0.985162	7.925	7.1 %	0.985161

Bảng 4. 1: Bảng kết quả chỉ số của mô hình

4.2. Hạn chế:

- Hạng giá vé "Economy" và "Business" chưa được phân ra để phân tích và dự báo độc lập, ảnh hưởng đến độ khách quan của kết quả
- Chia dữ liệu train-test chỉ 1 lần, có khả năng bỏ qua các điểm dữ liệu quan trọng ảnh hưởng đến kết quả.

4.3. Phương hướng phát triển:

- Tiếp tục thực hiện kết hợp Gradient Boosting với các thuật toán hiện có để khắc phục nhược điểm của thuật toán hiện tại và tăng độ chính xác của mô hình dự báo.
- Tìm hiểu và áp dụng K-Fold validation để đánh giá mô hình hiệu quả hơn và chọn ra được mô hình tốt nhất cho một bài toán thay vì chỉ đánh giá mô hình dựa trên một train set và một test set.
- Thu thập thêm dữ liệu để tăng tính khách quan, có thêm nhiều mẫu để thuật toán học tốt hơn.

TRÍCH DẪN TÀI LIỆU THAM KHẢO

[1] Victor E. Lee, Ruoming Jin, Lin Liu. “Chapter 4: Decision Trees: Theory and Algorithms”. [Online]. Nhận từ: http://www.odbms.org/wp-content/uploads/2014/07/DecisionTrees.pdf. Ngày truy cập: 30/10/2022.

[2] IBM. “What is a Decision Tree”. [Online]. Nhận từ: https://www.ibm.com/topics/decision-trees#:~:text=A%20decision%20tree%20is%20a,internal%20nodes%20and%20leaf%20nodes. Ngày truy cập: 30/10/2022.

[3] Hanady Abdulsalam, David B. Skillicorn, and Patrick Martin (2011). Classification using streaming random forests. IEEE Transactions on Knowledge and Data Engineering, 23(1), 22–36

[4] Trí tuệ nhân tạo (2019). "Cây Quyết Định (Decision Tree)". [Online]. Nhận từ: https://trituenhantao.io/kien-thuc/decision-tree/. Ngày truy cập: 31/10/2022.

[5] Sruthi E R (2021 - Cập nhật mới nhất: 06/2022). “Random Forest | Introduction to Random Forest Algorithm”. [Online]. Nhận từ: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/. Ngày truy cập 30/10/2022.

[6] Geeksforgeeks (Cập nhật mới nhất: 06/2022). “Bagging vs Boosting in Machine Learning”. [Online]. Nhận từ: https://www.geeksforgeeks.org/bagging-vs-boosting-in-machine-learning/. Ngày truy cập 2/11/2022.

[7] Nguyen Duy Sim (23/11/2018). “# Phân lớp bằng Random Forests trong Python”. [Online]. Nhận từ: https://viblo.asia/p/phan-lop-bang-random-forests-trong-python-djeZ1D2QKWz. Ngày truy cập 2/11/2022.

[8] Tavish Srivastava (2015 - Cập nhật mới nhất: 2020). “Tuning the parameters of your Random Forest model”. [Online]. Nhận từ: https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/. Ngày truy cập 3/11/2022.

[9] Fix, E. & Hodges, J.L. (1951) “Nonparametric Discrimination: Consistency Properties”, Randolph Field, Texas, Project 21-49-004, Report No. 4.

[10] Cover, T.M. & Hart, P.E. (1967) "Nearest neighbor pattern classification", IEEE Trans. Inf. Theory, 13: 21–27.

[11] S B Imandoust et al. (2013) “Journal of Engineering Research and Applications” Vol. 3, Issue 5, pp.605-610

[12] Onel Harrison (11/11/2018) “Machine Learning Basics with the K-Nearest Neighbors Algorithm” [Online]. Nhận từ: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761. Truy cập ngày: 2/11/2022.

[13] IBM. “What is the k-nearest neighbors algorithm?” [Online]. Nhận từ: https://www.ibm.com/topics/knn. Ngày truy cập: 2/11/2022.

[14] Machine Learning Mastery. “Machine Learning Mastery” [Online]. Nhận từ: https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/. Ngày truy cập: 1/11/2022.

[15] India Brand Equity Foundation. “Indian Aviation Industry”. [Online]. Nhận từ: https://www.ibef.org/industry/indian-aviation. Ngày truy cập: 25/10/2022.

[16] Kaki Raajitha, Meenakshi Kollati, Y.Mareswara Rao (2021). “Design of Thermometer Coding and One-Hot Coding” ICICV, Doi: 10.1109/ICICV50876.2021.9388570