# **LỜI CAM ĐOAN**

Chúng tôi xin cam kết đồ án cuối kì này là đề tài do chính chúng tôi nghiên cứu và thực hiện. Chúng tôi chỉ dựa trên mà không sao chép từ bất kì tài liệu hay công trình nào đã được thực hiện trước đó. Nếu có bất kỳ vi phạm nào, chúng tôi xin hoàn toàn chịu trách nhiệm.

Người thực hiện

**Nguyễn Đặng Mai Thy Trần Minh Đô**

**Mục lục**

[**LỜI CAM ĐOAN** 1](#_Toc138803957)

[**LỜI CẢM ƠN** 5](#_Toc138803958)

[**CHƯƠNG 1: TỔNG QUAN** 7](#_Toc138803959)

[1.1 ĐẶT VẤN ĐỀ 7](#_Toc138803960)

[1.2 MỤC TIÊU ĐỀ TÀI 7](#_Toc138803961)

[1.3 NỘI DUNG THỰC HIỆN 8](#_Toc138803962)

[1.4 KẾT QUẢ ĐẠT ĐƯỢC 10](#_Toc138803963)

[1.5 BỐ CỤC CỦA BÁO CÁO 11](#_Toc138803964)

[**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT** 12](#_Toc138803965)

[2.1 TỔNG QUAN VỀ KHOA HỌC DỮ LIỆU 12](#_Toc138803967)

[2.2 TỔNG QUAN VỀ LÝ THUYẾT HỌC MÁY (MACHINE LEARNING) 13](#_Toc138803968)

[2.3 TỔNG QUAN VỀ PHÂN LOẠI DỮ LIỆU 14](#_Toc138803969)

[2.4 TỔNG QUAN VỀ MẠNG LƯỚI NEURON NHÂN TẠO (ANN) 15](#_Toc138803970)

[**CHƯƠNG 3: CÔNG CỤ SỬ DỤNG** 17](#_Toc138803971)

[3.1 CÁC CÔNG CỤ ĐỂ CÀI ĐẶT MÔ PHỎNG TÍNH TOÁN 17](#_Toc138803972)

[3.2 PHẦN MỀM MÔ PHỎNG GOOGLE COLAB 18](#_Toc138803973)

[**CHƯƠNG 4: THIẾT KẾ VÀ MÔ HÌNH HOẠT ĐỘNG** 20](#_Toc138803974)

[4.1 DỮ LIỆU ĐẦU VÀO 20](#_Toc138803975)

[4.2 CÀI ĐẶT ỨNG DỤNG 22](#_Toc138803976)

[4.3 CÀI ĐẶT CÁC THƯ VIỆN 23](#_Toc138803977)

[4.4 CÁC MÔ HÌNH HỌC MÁY 26](#_Toc138803978)

[4.5 XÂY DỰNG THÀNH PHẦN 27](#_Toc138803979)

[**CHƯƠNG 5: KẾT QUẢ VÀ ĐÁNH GIÁ** 54](#_Toc138803980)

[5.1 PIPELINE TRỰC QUAN HÓA SỰ CHUYỂN ĐỔI 54](#_Toc138803981)

[5.2 TIỀN XỬ LÝ DỮ LIỆU 54](#_Toc138803982)

[5.3 ĐỒ THỊ 55](#_Toc138803983)

[5.4 MÔ HÌNH ĐỊNH GIÁ VÉ MÁY BAY THEO YÊU CẦU NGƯỜI DÙNG 77](#_Toc138803999)

[5.5 CÁC MÔ HÌNH DỰ ĐOÁN LIÊN QUAN 81](#_Toc138804000)

[**CHƯƠNG 6: KẾT LUẬN VÀ ĐỊNH HƯỚNG PHÁT TRIỂN** 86](#_Toc138804001)

[6.1 KẾT LUẬN 86](#_Toc138804002)

[6.2 ĐỊNH HƯỚNG PHÁT TRIỂN 86](#_Toc138804003)

[**CHƯƠNG 7: NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN VÀ TIẾN HÀNH VẤN ĐÁP VỀ ĐỀ TÀI** 88](#_Toc138804004)

[**TÀI LIỆU THAM KHẢO** 89](#_Toc138804005)

# **LỜI CẢM ƠN**

Lời đầu tiên chúng em xin cảm ơn Thầy Trương Quang Phúc đã đồng hành và hỗ trợ chúng em trong suốt học kì vừa qua, mặc dù chúng em chỉ mới gặp thầy trong một vài tháng nhưng chúng em đã cảm nhật được sự tận tình của thầy trong việc chỉ bảo, định hướng và hỗ trợ cho chúng em cả về vật chất và tình thần.

Đề tài của chúng em được hoàn thành trong 8 tuần, nó không phải là một thời gian dài nhưng đối với chúng em thì nó cũng khá gấp rút để hoàn thành đề tài vì khả năng và sự hiểu biết còn hạn hẹp. Chắc chắn đề tài chúng em sẽ không được như mong đợi của em và thầy nhưng đó là sự nỗ lực của chúng em trong quá trình vừa qua.

Chúng em mong rằng sẽ nhận được sự góp ý, bổ sung để đề tài của chúng em được hoàn thiện nhất có thể và rút ra những kinh nghiệm cho những môn học tiếp theo.

Một lần nữa chúng em xin cảm ơn thầy. Chúc thầy thật nhiều sức khỏe và luôn thành công trong công cuộc giảng dạy của mình.

**TÓM TẮT**

Đề tài "Định giá vé máy bay theo lịch trình và loại bay" tập trung vào việc áp dụng kỹ thuật khoa học dữ liệu để phân tích và định giá vé máy bay dựa trên các yếu tố như lịch trình và loại chuyến bay. Mục tiêu của đề tài là tạo ra một mô hình hoặc thuật toán dự đoán giá vé máy bay chính xác hơn dựa trên dữ liệu có sẵn.

Để thực hiện điều này, người nghiên cứu sẽ thu thập và xử lý dữ liệu về lịch trình và loại chuyến bay từ các nguồn khác nhau như hãng hàng không, trang web đặt vé, hay dữ liệu công khai từ các nguồn tin tức và diễn đàn. Dữ liệu này có thể bao gồm thông tin về ngày giờ bay, điểm khởi hành và điểm đến, loại chuyến bay (trực tiếp, chuyển đổi, chuyến bay nội địa, quốc tế, vv.), cũng như các yếu tố khác có thể ảnh hưởng đến giá vé như thời gian mua vé trước ngày bay, mùa du lịch, sự kiện đặc biệt, và tỷ lệ phòng vé còn lại.

Sau khi thu thập và tiền xử lý dữ liệu, người nghiên cứu sẽ áp dụng các phương pháp và thuật toán trong lĩnh vực khoa học dữ liệu như học máy, khai phá dữ liệu, và phân tích dữ liệu để xây dựng mô hình dự đoán giá vé máy bay. Qua đó, mô hình sẽ học từ dữ liệu huấn luyện có sẵn và dự đoán giá vé cho các lịch trình và loại chuyến bay mới.

Kết quả của đề tài sẽ là một mô hình hoặc thuật toán có khả năng dự đoán giá vé máy bay theo lịch trình và loại chuyến bay với độ chính xác cao. Mô hình này có thể được ứng dụng trong các công cụ tìm kiếm vé máy bay trực tuyến, hỗ trợ người dùng tìm kiếm vé máy bay với giá tốt hơn và tối ưu hóa quyết định mua vé.

# **CHƯƠNG 1: TỔNG QUAN**

## ĐẶT VẤN ĐỀ

Trong ngành hàng không, giá vé máy bay thường biến đổi theo nhiều yếu tố khác nhau như lịch trình, loại chuyến bay, mùa du lịch, sự kiện đặc biệt và sự cạnh tranh giữa các hãng hàng không. Việc định giá vé máy bay chính xác và hiệu quả là một thách thức đối với hãng hàng không và khách hàng.

Hiện nay, các hãng hàng không thường sử dụng các mô hình định giá phức tạp dựa trên quy tắc quản lý doanh thu (revenue management) để quản lý giá vé. Tuy nhiên, việc dự đoán và định giá chính xác các yếu tố như lịch trình và loại chuyến bay vẫn là một thách thức. Điều này có thể dẫn đến tình trạng giá vé không phản ánh đúng sự thay đổi của nhu cầu và cung cầu trên thị trường.

Vì vậy, để giải quyết vấn đề này, việc áp dụng kỹ thuật khoa học dữ liệu trong việc định giá vé máy bay theo lịch trình và loại chuyến bay là cần thiết. Bằng cách thu thập và phân tích dữ liệu về lịch trình, loại chuyến bay và các yếu tố liên quan, ta có thể xây dựng mô hình dự đoán giá vé máy bay chính xác hơn. Mô hình này có thể giúp hãng hàng không tối ưu hóa quyết định định giá và giúp khách hàng tìm kiếm vé máy bay với giá tốt hơn.

Điều này không chỉ mang lại lợi ích kinh tế cho hãng hàng không mà còn giúp cải thiện trải nghiệm của khách hàng, giảm sự phiền toái khi tìm kiếm vé máy bay với giá hợp lý và tăng tính minh bạch trong quá trình định giá vé máy bay.

## MỤC TIÊU ĐỀ TÀI

Mục tiêu của đề tài "Định giá vé máy bay theo lịch trình và loại bay" là xây dựng một mô hình hoặc thuật toán dự đoán giá vé máy bay chính xác hơn dựa trên dữ liệu về lịch trình và loại chuyến bay. Các mục tiêu cụ thể bao gồm:

Thu thập dữ liệu: Tiến hành thu thập dữ liệu về lịch trình và loại chuyến bay từ các nguồn khác nhau như hãng hàng không, trang web đặt vé, dữ liệu công khai và các nguồn tin tức có liên quan. Đảm bảo dữ liệu thu thập đủ phong phú và đáng tin cậy để xây dựng mô hình.

Tiền xử lý dữ liệu: Thực hiện quá trình tiền xử lý dữ liệu để chuẩn hóa, xử lý các giá trị thiếu, loại bỏ nhiễu và chuẩn bị dữ liệu cho việc xây dựng mô hình.

Xây dựng mô hình: Sử dụng các phương pháp và thuật toán trong lĩnh vực khoa học dữ liệu như học máy, khai phá dữ liệu và phân tích dữ liệu để xây dựng mô hình dự đoán giá vé máy bay. Mô hình sẽ học từ dữ liệu huấn luyện có sẵn và dự đoán giá vé cho các lịch trình và loại chuyến bay mới.

Đánh giá và tối ưu hóa mô hình: Đánh giá hiệu suất của mô hình thông qua các phương pháp đánh giá chất lượng dự đoán. Tối ưu hóa mô hình bằng cách điều chỉnh tham số, sử dụng kỹ thuật tăng cường dữ liệu và áp dụng các phương pháp tối ưu hóa khác.

Ứng dụng và thử nghiệm: Áp dụng mô hình vào thực tế thông qua việc tích hợp vào các công cụ tìm kiếm vé máy bay trực tuyến hoặc các hệ thống đặt vé máy bay. Thử nghiệm và đánh giá tính ứng dụng và hiệu quả của mô hình trong việc định giá vé máy bay theo lịch trình và loại chuyến bay.

Mục tiêu chính của đề tài là tạo ra một mô hình hoặc thuật toán có khả năng dự đoán giá vé máy bay chính xác hơn, giúp tối ưu hóa quyết định định giá và cung cấp cho khách hàng thông tin vé máy bay với giá tốt nhất dựa trên lịch trình và loại chuyến bay.

## NỘI DUNG THỰC HIỆN

Để thực hiện đề tài "Định giá vé máy bay theo lịch trình và loại bay" có thể thực hiện các bước sau:

1. Thu thập dữ liệu: Thu thập dữ liệu về lịch trình và loại chuyến bay từ các nguồn khác nhau như hãng hàng không, trang web đặt vé, dữ liệu công khai và các nguồn tin tức có liên quan. Cần xác định các yếu tố quan trọng trong việc định giá vé máy bay và thu thập thông tin tương ứng.

2. Tiền xử lý dữ liệu: Thực hiện quá trình tiền xử lý dữ liệu để chuẩn hóa và chuẩn bị dữ liệu cho việc xây dựng mô hình. Bước này bao gồm loại bỏ dữ liệu không hợp lệ hoặc thiếu, xử lý các giá trị ngoại lệ (outliers), chuẩn hóa đơn vị đo lường, và mã hóa các biến phân loại thành dạng số nếu cần thiết.

3. Phân tích và khám phá dữ liệu: Áp dụng các phương pháp khai phá dữ liệu và phân tích dữ liệu để tìm hiểu mối quan hệ giữa lịch trình, loại chuyến bay và giá vé máy bay. Có thể sử dụng các phương pháp như phân tích hồi quy, phân tích chuỗi thời gian, phân tích yếu tố quyết định giá cả, và các kỹ thuật trực quan hóa dữ liệu để khám phá thông tin quan trọng và hiểu rõ hơn về dữ liệu.

4. Xây dựng mô hình dự đoán: Sử dụng các phương pháp học máy và thuật toán trong lĩnh vực khoa học dữ liệu để xây dựng mô hình dự đoán giá vé máy bay. Có thể áp dụng các phương pháp như hồi quy tuyến tính, hồi quy logistic, máy vector hỗ trợ (SVM), mạng nơ-ron (neural networks), hoặc các mô hình dự đoán khác phù hợp với bài toán.

5. Đánh giá mô hình: Đánh giá hiệu suất của mô hình thông qua các phương pháp đánh giá chất lượng dự đoán như sai số trung bình (mean squared error), hệ số xác định (coefficient of determination), hay độ chính xác (accuracy) (tuỳ thuộc vào bài toán). Đồng thời, cần kiểm tra tính ổn định và độ tin cậy của mô hình bằng cách sử dụng phương pháp chia tách dữ liệu (cross-validation) hoặc chia tách dữ liệu theo thời gian.

6. Tối ưu hóa và điều chỉnh mô hình: Dựa trên kết quả đánh giá, tối ưu hóa mô hình bằng cách điều chỉnh tham số hoặc áp dụng các phương pháp tối ưu hóa khác như lựa chọn biến quan trọng, tăng cường dữ liệu (data augmentation), hoặc sử dụng các kỹ thuật tối ưu hóa mạng nơ-ron.

7. Thử nghiệm và ứng dụng: Áp dụng mô hình vào thực tế thông qua việc tích hợp vào các công cụ tìm kiếm vé máy bay trực tuyến hoặc các hệ thống đặt vé máy bay. Thử nghiệm và đánh giá tính ứng dụng và hiệu quả của mô hình trong việc định giá vé máy bay theo lịch trình và loại chuyến bay.

Trên cơ sở các bước trên, đề tài sẽ tiến hành thực hiện để đạt được mục tiêu xây dựng mô hình dự đoán giá vé máy bay chính xác hơn dựa trên lịch trình và loại chuyến bay.

## KẾT QUẢ ĐẠT ĐƯỢC

Kết quả đạt được từ đề tài "Định giá vé máy bay theo lịch trình và loại bay" có thể bao gồm:

1. Mô hình dự đoán giá vé máy bay chính xác hơn: Kết quả đáng chú ý là việc xây dựng một mô hình dự đoán giá vé máy bay dựa trên lịch trình và loại chuyến bay. Mô hình này có khả năng dự đoán giá vé với độ chính xác cao hơn so với các phương pháp truyền thống.

2. Tối ưu hóa quyết định định giá: Kết quả đề tài có thể giúp hãng hàng không tối ưu hóa quyết định định giá vé máy bay dựa trên lịch trình và loại chuyến bay. Thông qua mô hình dự đoán, hãng hàng không có thể điều chỉnh giá vé để tăng cường doanh thu và cạnh tranh trên thị trường.

3. Cải thiện trải nghiệm khách hàng: Kết quả từ đề tài có thể giúp khách hàng tìm kiếm và chọn vé máy bay với giá tốt hơn dựa trên lịch trình và loại chuyến bay. Điều này giúp cải thiện trải nghiệm mua vé của khách hàng và giảm sự phiền toái trong việc tìm kiếm giá vé hợp lý.

4. Tăng tính minh bạch trong quá trình định giá vé máy bay: Kết quả đề tài cung cấp một phương pháp định giá vé máy bay dựa trên dữ liệu và khoa học dữ liệu, tăng tính minh bạch và đáng tin cậy trong quá trình định giá vé. Điều này giúp khách hàng và các bên liên quan hiểu rõ hơn về cơ sở định giá và cách giá vé được hình thành.

5. Đóng góp cho lĩnh vực nghiên cứu và ứng dụng: Kết quả từ đề tài có thể đóng góp cho lĩnh vực nghiên cứu về định giá vé máy bay và ứng dụng trong thực tế. Các phương pháp và kỹ thuật trong đề tài có thể áp dụng rộng rãi trong ngành hàng không và các lĩnh vực liên quan khác.

Tổng quan, kết quả từ đề tài có thể mang lại lợi ích kinh tế cho hãng hàng không, cải thiện trải nghiệm của khách hàng và góp phần phát triển trong lĩnh vực định giá vé máy bay theo lịch trình và loại chuyến bay.

## BỐ CỤC CỦA BÁO CÁO

**Chương 1:** Tổng quan đề tài

Nêu những tổng quan về đề tài để giáo viên có cái nhìn chung nhất về dự án trước khi đến với chi tiết. Bên cạnh đó nếu ra mục tiêu, giới hạn của đề tài để hiểu sâu hơn về dự án

**Chương 2:** Cơ sở lý thuyết

Với mỗi bài toán được đặt ra đều dựa trên những kiến thức nền tảng. Chương 2 sẽ nêu ra những lý thuyết cơ sở để hoàn thành được đề tài này.

**Chương 3:** Công cụ sử dụng

Nêu ra những công cụ được sử dụng cho đề tài, trình bày các phần mềm đã được sử dụng để lập trình, lưu trữ và điều khiển các thiết bị vật lý đã được trình bày trước đó.

**Chương 4:** Thiết kế và mô hình hoạt động

Dựa vào những kiến thức đề cập ở trên, tiến hành thiết kế hệ thống. Trình bày tổng quan hệ thống, sơ đồ khối hoạt động chung.

**Chương 5:** Kết quả và đánh giá

Cung cấp hình ảnh, mã code, mô hình thực tế liên quan và phân tích về kết quả thực nghiệm khi vận hành của đề tài. Nêu ra những khó khăn gặp phải khi thực nghiệm cũng như báo cáo về những phần chưa làm được.

**Chương 6:** Kết luận và định hướng phát triển sản phẩm trong tương lai

Đưa ra nhận xét về tiến trình, tiến độ hoàn thành đề tài cũng như đánh giá các ưu, nhược điểm của hệ thống. Xem xét độ khả khi của đề tài, đảm bảo rằng đề tài có khả năng hoàn thiện trong tương lai. Qua đó nêu ra các ý tưởng để cải thiện và phát triển đề tài sau này.

**Chương 7:** Lời nhận xét và giáo viên tiến hành vấn đáp chi tiết về đề tài.

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**



## TỔNG QUAN VỀ KHOA HỌC DỮ LIỆU

Khoa học dữ liệu là lĩnh vực nghiên cứu và ứng dụng các phương pháp, công cụ và kỹ thuật để khai thác, hiểu và rút trích thông tin từ dữ liệu. Nó kết hợp các lĩnh vực như xử lý dữ liệu, học máy, thống kê, khai phá dữ liệu và trực quan hóa dữ liệu để phân tích và tạo ra thông tin hữu ích từ dữ liệu.

Lĩnh vực khoa học dữ liệu có vai trò quan trọng trong việc giải quyết các vấn đề phức tạp và đưa ra quyết định dựa trên dữ liệu. Các ứng dụng của khoa học dữ liệu rất đa dạng và tồn tại trong nhiều lĩnh vực, bao gồm kinh doanh, y tế, tài chính, marketing, giao thông vận tải, xử lý ngôn ngữ tự nhiên và nhiều lĩnh vực khác.

Quá trình làm việc trong khoa học dữ liệu thường bao gồm các bước sau:

1. Thu thập dữ liệu: Bước này liên quan đến việc thu thập dữ liệu từ các nguồn khác nhau như cơ sở dữ liệu, tệp tin, mạng xã hội, cảm biến và nhiều nguồn dữ liệu khác.

2. Tiền xử lý dữ liệu: Dữ liệu thu thập được thường không hoàn hảo, chứa nhiễu hoặc thiếu thông tin. Bước tiền xử lý dữ liệu bao gồm làm sạch, chuẩn hóa, loại bỏ dữ liệu nhiễu, xử lý giá trị thiếu và biến đổi dữ liệu thành dạng phù hợp cho việc phân tích.

3. Khám phá dữ liệu: Bước này tìm hiểu dữ liệu thông qua việc áp dụng các phương pháp và kỹ thuật khai phá dữ liệu để tìm ra mô hình, quy luật, xu hướng hoặc thông tin quan trọng trong dữ liệu.

4. Xây dựng mô hình: Dựa trên dữ liệu đã được tiền xử lý và khám phá, xây dựng mô hình học máy hoặc thống kê để dự đoán, phân loại hoặc mô tả dữ liệu. Các phương pháp phổ biến bao gồm hồi quy tuyến tính, cây quyết định, máy vector hỗ trợ, mạng nơ-ron, và các phương pháp học sâu khác.

5. Đánh giá và tối ưu hóa: Đánh giá mô hình dựa trên các phép đo độ chính xác, độ tin cậy hoặc các yếu tố khác tùy thuộc vào mục tiêu của vấn đề. Tối ưu hóa mô hình bằng cách điều chỉnh tham số hoặc áp dụng các phương pháp tối ưu hóa khác như chia tách dữ liệu và kiểm định chéo.

6. Triển khai và ứng dụng: Khi mô hình đã được đánh giá và tối ưu, nó có thể được triển khai và ứng dụng vào thực tế. Điều này bao gồm việc tích hợp mô hình vào hệ thống, ứng dụng hoặc quy trình để đưa ra quyết định dựa trên dữ liệu.

Khoa học dữ liệu cung cấp những cơ hội đáng kể trong việc tận dụng thông tin từ dữ liệu và giải quyết các vấn đề phức tạp. Tuy nhiên, cần chú ý đến các thách thức như bảo mật dữ liệu, đạo đức và quyền riêng tư, và sự hiểu biết đúng đắn về việc diễn giải và áp dụng kết quả từ khoa học dữ liệu.

## TỔNG QUAN VỀ LÝ THUYẾT HỌC MÁY (MACHINE LEARNING)

Lý thuyết học máy (machine learning) là một lĩnh vực trong trí tuệ nhân tạo (AI) và khoa học dữ liệu (data science) tập trung vào việc phát triển các phương pháp và thuật toán để máy tính có thể tự học từ dữ liệu và làm các dự đoán hoặc đưa ra quyết định mà không cần được lập trình cụ thể.

Trong lý thuyết học máy, một hệ thống máy tính được huấn luyện từ dữ liệu để xác định các mẫu, xu hướng và quy luật trong dữ liệu đó. Các thuật toán học máy phân loại thành hai loại chính:

1. Học có giám sát (supervised learning): Trong học có giám sát, dữ liệu huấn luyện được cung cấp với các cặp đầu vào và đầu ra tương ứng. Mục tiêu là xây dựng một mô hình học máy từ dữ liệu huấn luyện này để có thể dự đoán đầu ra cho các đầu vào mới. Các thuật toán phổ biến trong học có giám sát bao gồm hồi quy tuyến tính, cây quyết định, máy vector hỗ trợ (SVM), và mạng nơ-ron nhân tạo.

2. Học không giám sát (unsupervised learning): Trái ngược với học có giám sát, trong học không giám sát, không có đầu ra đã biết trước được cung cấp trong quá trình huấn luyện. Mục tiêu là tìm ra cấu trúc, mô hình, hoặc nhóm của dữ liệu. Các thuật toán phổ biến trong học không giám sát bao gồm phân cụm (clustering), phân tích thành phần chính (PCA), và phân tích tiếp cận gần (nearest neighbor).

Lý thuyết học máy cũng bao gồm các khái niệm và công cụ liên quan như cross-validation (kiểm định chéo), regularization (chính quy hóa), feature selection (lựa chọn đặc trưng), và ensemble learning (học kết hợp). Ngoài ra, các phương pháp học máy tiên tiến như học sâu (deep learning) và học tăng cường (reinforcement learning) cũng là những lĩnh vực đang phát triển mạnh mẽ trong học máy.

Lý thuyết học máy cung cấp cơ sở toán học và thuật toán cho việc xây dựng và ứng dụng các mô hình học máy trong nhiều lĩnh vực, bao gồm nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên, dự báo tài chính, và nhiều ứng dụng khác. Nó đã đóng vai trò quan trọng trong sự phát triển của trí tuệ nhân tạo và công nghệ thông tin.

## TỔNG QUAN VỀ PHÂN LOẠI DỮ LIỆU

Phân loại dữ liệu là quá trình phân chia các mẫu dữ liệu vào các nhóm hoặc lớp khác nhau dựa trên các đặc trưng hay thuộc tính của chúng. Đây là một trong những tác vụ quan trọng trong lĩnh vực học máy và khai phá dữ liệu.

Quá trình phân loại dữ liệu thường bao gồm các bước sau:

1. Thu thập dữ liệu: Bước này đảm bảo có đủ dữ liệu để huấn luyện và xây dựng mô hình phân loại. Dữ liệu có thể được thu thập từ nhiều nguồn khác nhau, bao gồm cơ sở dữ liệu, tệp tin, trang web, và nhiều nguồn khác.

2. Tiền xử lý dữ liệu: Trong bước này, dữ liệu được làm sạch, chuẩn hóa và tiền xử lý để loại bỏ dữ liệu nhiễu, xử lý giá trị thiếu, và chuyển đổi dữ liệu về định dạng phù hợp cho quá trình phân loại.

3. Xác định đặc trưng: Đặc trưng của dữ liệu được chọn hoặc trích xuất từ dữ liệu gốc. Điều này có thể bao gồm các thuật toán tiền xử lý dữ liệu, trích xuất đặc trưng từ ảnh, hoặc xử lý ngôn ngữ tự nhiên để rút trích thông tin quan trọng.

4. Xây dựng và huấn luyện mô hình: Một mô hình học máy hoặc thuật toán phân loại được xây dựng và huấn luyện trên dữ liệu huấn luyện. Mô hình này sẽ học từ các mẫu dữ liệu đã được gán nhãn và cố gắng tìm hiểu các quy tắc hoặc mối quan hệ giữa đặc trưng và nhãn.

5. Đánh giá và tinh chỉnh mô hình: Mô hình được đánh giá bằng cách sử dụng các phép đo độ chính xác, độ tin cậy hoặc các yếu tố khác tùy thuộc vào bài toán phân loại cụ thể. Nếu cần thiết, mô hình có thể được tinh chỉnh bằng cách điều chỉnh tham số hoặc áp dụng các phương pháp tối ưu hóa.

6. Sử dụng mô hình phân loại: Sau khi mô hình đã được đánh giá và tinh chỉnh, nó có thể được sử dụng để phân loại các mẫu dữ liệu mới. Dựa vào đặc trưng của mẫu dữ liệu đầu vào, mô hình sẽ dự đoán và gán nhãn cho mẫu đó thuộc vào nhóm hay lớp nào.

Phân loại dữ liệu có nhiều ứng dụng thực tế, bao gồm phân loại email rác, phát hiện gian lận tín dụng, nhận dạng khuôn mặt, và nhiều lĩnh vực khác. Các thuật toán phân loại phổ biến bao gồm cây quyết định, hỗn hợp Gaussian, máy vector hỗ trợ (SVM), và mạng nơ-ron nhân tạo.

## TỔNG QUAN VỀ MẠNG LƯỚI NEURON NHÂN TẠO (ANN)

Mạng lưới neuron nhân tạo (Artificial Neural Network - ANN) là một mô hình tính toán được lấy cảm hứng từ cấu trúc và hoạt động của hệ thống thần kinh sinh học. ANN là một tập hợp các đơn vị tính toán gọi là neuron nhân tạo, được tổ chức thành các lớp và liên kết với nhau thông qua các trọng số.

Cấu trúc chính của một ANN bao gồm:

- Lớp đầu vào (input layer): Nhận các giá trị đầu vào và truyền chúng đến các neuron trong lớp tiếp theo.

- Các lớp ẩn (hidden layers): Là các lớp nằm giữa lớp đầu vào và lớp đầu ra. Các neuron trong các lớp ẩn tính toán và truyền giá trị đến các neuron trong lớp tiếp theo.

- Lớp đầu ra (output layer): Tạo ra kết quả cuối cùng của mạng dựa trên thông tin từ các lớp trước đó.

Mỗi neuron trong mạng lưới nhân tạo tính toán dựa trên nguyên tắc của một neuron sinh học. Nó nhận đầu vào từ các neuron trong lớp trước đó, thực hiện một phép tính tuyến tính và áp dụng một hàm kích hoạt phi tuyến để tạo ra đầu ra. Các trọng số trong mạng lưới nhân tạo được sử dụng để điều chỉnh tầng đầu vào và tạo ra kết quả dự đoán chính xác.

Quá trình huấn luyện một ANN thường bao gồm:

1. Khởi tạo trọng số: Các trọng số ban đầu của mạng lưới được khởi tạo ngẫu nhiên.

2. Lan truyền thuận (forward propagation): Dữ liệu huấn luyện được đưa qua mạng để tính toán kết quả dự đoán.

3. So sánh kết quả: Kết quả dự đoán được so sánh với giá trị thực tế để tính toán lỗi.

4. Lan truyền ngược (backpropagation): Lỗi được truyền ngược từ lớp đầu ra đến lớp đầu vào để điều chỉnh trọng số theo hướng giảm thiểu lỗi.

5. Cập nhật trọng số: Trọng số được cập nhật dựa trên thuật toán tối ưu hóa như gradient descent để giảm thiểu lỗi.

Mạng lưới neuron nhân tạo đã được áp dụng rộng rãi trong nhiều lĩnh vực, bao gồm nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên, dự báo tài chính, và các bài toán phân loại và dự đoán khác. Sự phát triển của ANN đã tạo ra các kiến trúc phức tạp hơn như mạng nơ-ron sâu (deep neural networks) và các thuật toán học sâu khác, làm cải thiện đáng kể hiệu suất và khả năng của hệ thống học máy.

# **CHƯƠNG 3: CÔNG CỤ SỬ DỤNG**

## 3.1 CÁC CÔNG CỤ ĐỂ CÀI ĐẶT MÔ PHỎNG TÍNH TOÁN

Python cung cấp nhiều công cụ tính toán mạnh mẽ để làm việc với dữ liệu số, phân tích dữ liệu và xây dựng mô hình học máy. Các công cụ quan trọng trong Python bao gồm NumPy, Pandas, TensorFlow và Keras.

NumPy: NumPy (Numerical Python) là một thư viện Python phổ biến được sử dụng cho tính toán số học và xử lý mảng đa chiều. Nó cung cấp các cấu trúc dữ liệu và hàm toán học mạnh mẽ cho việc làm việc với mảng và ma trận. NumPy là một phần quan trọng của hệ sinh thái Python cho khoa học dữ liệu và học máy.

Pandas: Pandas là một thư viện mạnh mẽ cho phân tích dữ liệu và xử lý dữ liệu có cấu trúc. Nó cung cấp các đối tượng dữ liệu như DataFrame, Series và các chức năng cho việc truy vấn, lọc, sắp xếp, tổng hợp và biến đổi dữ liệu. Pandas rất hữu ích cho việc khám phá và tiền xử lý dữ liệu trước khi xây dựng mô hình học máy.

TensorFlow: TensorFlow là một thư viện mã nguồn mở phổ biến trong lĩnh vực học máy và trí tuệ nhân tạo. Nó cung cấp một hệ thống khung làm việc linh hoạt cho việc xây dựng và huấn luyện mạng nơ-ron nhân tạo. TensorFlow hỗ trợ tính toán phân tán và tính toán trên GPU, cho phép xây dựng và huấn luyện các mô hình học sâu phức tạp.

Keras: Keras là một giao diện lập trình ứng dụng (API) cao cấp được xây dựng trên TensorFlow. Nó cung cấp các lớp, phương thức và hàm tiện ích để dễ dàng xây dựng, huấn luyện và đánh giá mô hình học sâu. Keras được đánh giá cao về tính đơn giản, linh hoạt và hiệu suất trong việc xây dựng các mô hình học máy.

Tổng quan về các công cụ này cho thấy Python là một ngôn ngữ lập trình phổ biến và mạnh mẽ trong lĩnh vực khoa học dữ liệu và học máy, cung cấp các thư viện và công cụ hỗ trợ cho việc tính toán, xử lý dữ liệu và xây dựng mô hình.

## 3.2 PHẦN MỀM MÔ PHỎNG GOOGLE COLAB

Google Colab là một môi trường mô phỏng trực tuyến miễn phí được cung cấp bởi Google. Nó cho phép bạn viết và chạy mã Python trong các "notebook" tương tự như Jupyter Notebook. Google Colab được thiết kế để hỗ trợ việc phát triển và chia sẻ dự án machine learning và deep learning.

Dưới đây là một số tính năng quan trọng của Google Colab:

* Môi trường Jupyter Notebook: Google Colab sử dụng một môi trường notebook tương tự như Jupyter Notebook. Điều này cho phép bạn viết và chạy mã Python theo từng ô (cell) riêng biệt.
* Hỗ trợ GPU và TPU: Google Colab cung cấp môi trường chạy trên các máy ảo được cung cấp bởi Google, bao gồm hỗ trợ GPU và TPU. Điều này cho phép bạn sử dụng khả năng tính toán mạnh mẽ của GPU và TPU để tăng tốc độ huấn luyện mô hình machine learning và deep learning.
* Thư viện và công cụ tích hợp: Google Colab hỗ trợ cài đặt và sử dụng các thư viện Python phổ biến như NumPy, Pandas, TensorFlow và PyTorch. Nó cũng cung cấp công cụ tích hợp như trình soạn thảo mã, gỡ rối và trình duyệt tệp tin để giúp bạn phát triển và kiểm tra mã một cách thuận tiện.
* Lưu trữ và chia sẻ: Google Colab cho phép bạn lưu trữ các notebook của mình trên Google Drive và chia sẻ chúng với người khác. Điều này giúp bạn lưu trữ dự án của mình và làm việc cùng nhóm một cách dễ dàng.
* Tích hợp dịch vụ điện toán đám mây: Google Colab tích hợp tốt với các dịch vụ điện toán đám mây khác của Google như Google Cloud Storage và BigQuery. Điều này cho phép bạn làm việc với dữ liệu lớn và triển khai mô hình của mình trên các tài nguyên điện toán mạnh mẽ của Google.

Google Colab là một công cụ mô phỏng mạnh mẽ và tiện lợi cho việc phát triển và chia sẻ dự án machine learning và deep learning. Nó cung cấp một môi trường trực tuyến dễ sử dụng và tích hợp nhiều tính năng hữu ích giúp nâng cao hiệu suất và tiện ích trong quá trình phát triển.

# **` CHƯƠNG 4: THIẾT KẾ VÀ MÔ HÌNH HOẠT ĐỘNG**

## 4.1 DỮ LIỆU ĐẦU VÀO

Tập dữ liệu được lấy từ trang web "Ease My Trip" - Một nền tảng internet đặt vé máy bay thông dụng. Dữ liệu được thu thập trong 50 ngày ( 11/02/2022 - 31/03/2022 ) chứa thông tin về các tùy chọn đặt chuyến bay giữa 6 thành phố lớn hàng đầu của Ấn Độ. Bao gồm 2 phần ( economy, business ), 300261 tùy chọn đặt vé và 11 tính năng.

Các tính năng bao gồm:

* Airline: Hãng hàng không - phân loại 6 hãng hàng không.
* Flight Code: Mã chuyến bay - phân loại chuyến bay.
* Source City: Nơi cất cánh - phân loại 6 thành phố.
* Departure Time: Thời gian khởi hành - phân loại 6 nhãn thời gian, dữ liệu thu được bằng cách nhóm các khoảng thời gian thành các bins.
* Number of Stops: Điểm dừng - phân loại 3 giá trị là số điểm dừng giữa nơi cất cánh và nơi hạ cánh.
* Arrival Time: Thời gian đến - phân loại 6 nhãn thời gian, dữ liệu thu được bằng cách nhóm các khoảng thời gian thành các bins.
* Destination City: Nơi hạ cánh - phân loại 6 thành phố.
* Journey Class: Hạng - phân loại 2 hạng ghế.
* Trip Duration: Thời lượng - liên tục hiển thị tổng thời gian bay ( tính bằng giờ ).
* Days Left: Số ngày còn lại - số ngày chênh lệch giữa ngày bay và ngày 10/02/2022.
* Ticket Price: Giá vé - biến mục tiêu lưu thông tin về giá vé
* Date of Journey: Ngày hành trình của chuyến bay - phân loại theo ngày tháng năm. ( Tính năng được bổ sung sau khi thực hiện tiền xử lý dữ liệu )
* Day of Week: Thời gian hành trình của chuyến bay - là thời gian ( thứ trong tuần ) của Date of Journey. ( Tính năng được bổ sung sau khi thực hiện tiền xử lý dữ liệu )

Dữ liệu đầu vào được lấy từ Kaggle là một tệp gồm 3 file csv.

Link datasheet: <https://www.kaggle.com/datasets/shubhambathwal/flight-price-prediction?select=economy.csv>

* Cách tải dữ liệu lên Google Colab

Để tải dữ liệu lên Google Colab, bạn có thể sử dụng một số phương pháp sau:

Tải dữ liệu từ URL: Nếu dữ liệu của bạn được lưu trữ trực tuyến, bạn có thể sử dụng lệnh `wget` để tải nó trực tiếp từ URL.

!wget <URL\_dữ\_liệu>

Tải dữ liệu từ Google Drive: Nếu dữ liệu của bạn được lưu trữ trong Google Drive, bạn có thể sử dụng thư viện `google.colab` để kết nối với Google Drive và tải dữ liệu từ đó.

from google.colab import drive

drive.mount('/content/drive')

Tải dữ liệu từ máy tính cá nhân: Nếu dữ liệu của bạn được lưu trữ trên máy tính cá nhân, bạn có thể sử dụng chức năng tải lên của Google Colab để tải dữ liệu từ máy tính cá nhân lên Colab. Bạn có thể sử dụng giao diện người dùng hoặc sử dụng câu lệnh sau:

from google.colab import files

uploaded = files.upload()

## 4.2 CÀI ĐẶT ỨNG DỤNG

Để sử dụng Google Colab, bạn có thể làm theo các bước sau:

1. Truy cập vào trang web của Google Colab: Để bắt đầu, hãy truy cập vào trang web Google Colab tại địa chỉ https://colab.research.google.com.
2. Tạo một notebook mới: Khi trang web Colab được tải, bạn sẽ thấy một danh sách các notebook đã được tạo hoặc có thể tạo mới. Bạn có thể tạo một notebook mới bằng cách nhấp vào nút "New Notebook" hoặc tệp tin > "New Python 3 notebook".
3. Lập trình trong notebook: Khi bạn đã tạo một notebook mới, bạn có thể bắt đầu lập trình bằng ngôn ngữ Python. Mỗi ô (cell) trong notebook là một đơn vị thực thi độc lập, cho phép bạn viết và chạy mã Python một cách tương tác.
4. Chạy mã trong notebook: Để chạy mã trong một ô, bạn có thể nhấp vào ô đó và sử dụng phím tắt Ctrl + Enter hoặc nhấp vào nút "Play" ở góc trái của ô. Colab sẽ thực thi mã và hiển thị kết quả trực tiếp dưới ô đó.
5. Thêm và xóa ô: Bạn có thể thêm một ô mới bằng cách nhấp vào nút "+ Code" hoặc "+ Text" ở góc trên bên trái của notebook. Để xóa một ô, bạn có thể nhấp vào biểu tượng thùng rác ở góc trên bên phải của ô đó.
6. Lưu và chia sẻ notebook: Colab tự động lưu các thay đổi trong notebook của bạn vào Google Drive. Bạn cũng có thể tải xuống notebook dưới dạng file .ipynb để lưu trữ hoặc chia sẻ với người khác.
7. Cài đặt thư viện và công cụ: Bạn có thể cài đặt các thư viện và công cụ Python bổ sung bằng cách sử dụng lệnh pip trong các ô code của Colab. Ví dụ: "!pip install numpy" để cài đặt thư viện NumPy.
8. Sử dụng GPU và TPU: Colab cung cấp tài nguyên GPU và TPU miễn phí cho việc chạy mã. Bạn có thể chọn "Runtime" trong menu và chọn "Change runtime type" để chọn GPU hoặc TPU làm môi trường thực thi.

## 4.3 CÀI ĐẶT CÁC THƯ VIỆN

1. Pandas: là một thư viện mã nguồn mở dùng để xử lý và phân tích dữ liệu dạng bảng. Nó cung cấp các công cụ để đọc, ghi, lọc, biến đổi và thống kê dữ liệu một cách dễ dàng.
2. Numpy: (Numerical Python) là một thư viện mã nguồn mở dùng để làm việc với mảng và ma trận số học. Nó cung cấp các công cụ toán học và thao tác mảng hiệu quả.
3. Seaborn: là một thư viện mã nguồn mở dùng để trực quan hóa dữ liệu. Nó cung cấp các công cụ và giao diện đơn giản để tạo ra các biểu đồ thống kê và biểu đồ phân phối hấp dẫn.
4. Keras: là một thư viện mã nguồn mở cho việc xây dựng và huấn luyện các mạng neural network. Keras hỗ trợ nhiều backend như TensorFlow, Theano và CNTK.

* Keras.models cung cấp các lớp và công cụ để xây dựng các mô hình học sâu như mạng nơ-ron nhân tạo (Artificial Neural Networks - ANN) và mạng hồi quy (Recurrent Neural Networks - RNN).
* Keras.layers cung cấp các lớp khác nhau để xây dựng các kiến trúc mô hình học sâu, từ các lớp cơ bản như Dense, Conv2D, LSTM đến các lớp đặc biệt như Dropout, BatchNormalization, và nhiều lớp khác.

1. Matplotlib: là một thư viện mã nguồn mở dùng để trực quan hóa dữ liệu. Nó cung cấp các công cụ và chức năng để tạo và tùy chỉnh các biểu đồ và đồ thị.

* Matplotlib.pyplot là một module trong thư viện Matplotlib dùng để vẽ và tùy chỉnh các biểu đồ trong Python. Nó cung cấp các chức năng để tạo các biểu đồ đơn giản và điều chỉnh các yếu tố của chúng.

1. Datetime: là một trong những thư viện cốt lõi trong Python và được sử dụng để làm việc với các đối tượng ngày tháng và thời gian. Nó cung cấp các lớp và phương thức để thao tác, định dạng và tính toán các giá trị ngày tháng.
2. Statsmodels: là một thư viện mạnh mẽ để thực hiện các phân tích thống kê và mô hình hóa dữ liệu. Nó cung cấp các công cụ và chức năng cho việc thực hiện các phân tích tương quan, hồi quy, mô phỏng dữ liệu, kiểm định giả thuyết và nhiều hơn nữa.
3. Statsmodels.api cung cấp nhiều chức năng và phương pháp phân tích thống kê khác nhau, cho phép bạn thực hiện nhiều loại phân tích phức tạp.
4. Scipy: là một thư viện Python sử dụng cho tính toán khoa học và tính toán số học. Nó cung cấp các công cụ và chức năng cho tối ưu hóa, tích phân, đại số tuyến tính, xử lý tín hiệu, xử lý hình ảnh, và nhiều lĩnh vực khác liên quan đến khoa học dữ liệu.

* Scipy.stats cung cấp nhiều chức năng và lớp hữu ích cho các tác vụ thống kê và phân tích dữ liệu, và nó là một công cụ quan trọng trong lĩnh vực khoa học dữ liệu và thống kê.

1. Math: là một thư viện trong Python cung cấp các hàm toán học cơ bản để thực hiện các phép tính và tính toán trên số học. Bạn có thể sử dụng math để thực hiện các phép tính như căn bậc hai, lũy thừa, logarit, trị tuyệt đối, các hàm trigonometric, và nhiều hàm toán học khác.
2. Xgboost: là một thư viện mã nguồn mở được sử dụng trong học máy và khai phá dữ liệu. Nó là một thuật toán học tổ hợp dựa trên gradient boosting và được sử dụng để xây dựng và tinh chỉnh các mô hình học máy mạnh mẽ và hiệu quả.
3. Catboost: là một thư viện mã nguồn mở trong học máy được thiết kế để làm việc tốt với dữ liệu có biến rời rạc. Nó tự động xử lý biến rời rạc và có hiệu suất và khả năng tổng quát hóa tốt trên các tập dữ liệu đa dạng. ( cần phải install - Google Colab chưa hỗ trợ sẵn )
4. Tensorflow: là một thư viện mã nguồn mở để xây dựng và huấn luyện các mô hình học máy và trí tuệ nhân tạo. Nó cung cấp một cấu trúc dữ liệu gọi là "graph" để mô tả mô hình và sử dụng các phép tính số học để điều chỉnh các tham số. TensorFlow hỗ trợ nhiều công cụ và tính năng mạnh mẽ để tạo ra và triển khai các mô hình học máy phức tạp.

* Tensorflow.keras.optimizers cung cấp các tối ưu hóa để điều chỉnh các tham số mạng nơ-ron trong quá trình huấn luyện. Bạn có thể sử dụng các tối ưu hóa như SGD, RMSprop, Adam, và nhiều thuật toán khác để cải thiện hiệu suất và tốc độ học của mô hình.

1. Sklearn: (Scikit-learn) là một thư viện mã nguồn mở trong Python được sử dụng phổ biến trong lĩnh vực học máy và khai phá dữ liệu. Sklearn cung cấp nhiều công cụ và thuật toán học máy để thực hiện các tác vụ như phân loại, hồi quy, gom cụm, và tiền xử lý dữ liệu.

* Sklearn.model\_selection được sử dụng để chia tập dữ liệu thành các tập huấn luyện và kiểm tra, và thực hiện các phương pháp đánh giá mô hình.
* Sklearn.metrics cung cấp các hàm để đánh giá hiệu suất của mô hình học máy.
* Sklearn.linear\_model chứa các mô hình hồi quy tuyến tính và lasso.
* Sklearn.preprocessing cung cấp các công cụ để tiền xử lý và chuẩn hóa dữ liệu.
* Sklearn.compose cung cấp các công cụ để xử lý dữ liệu độc lập cho từng cột và làm việc với đối tượng mục tiêu trong quá trình huấn luyện mô hình.
* Sklearn.ensemble cung cấp các công cụ để xây dựng và sử dụng các mô hình như random forest, AdaBoost và Gradient Boosting.
* Sklearn.tree bao gồm các mô hình cây quyết định và random forest.
* Sklearn.pipeline cung cấp các lớp quan trọng như Pipeline để xây dựng pipeline, và FeatureUnion để kết hợp các đặc trưng từ các transformer khác nhau.
* Sklearn.feature\_selection cung cấp các hàm và lớp để giúp xác định các đặc trưng quan trọng nhất trong một tập dữ liệu.

## 4.4 CÁC MÔ HÌNH HỌC MÁY

1. LinearRegression là một thuật toán học máy sử dụng để tìm mối quan hệ tuyến tính giữa các biến đầu vào và biến đầu ra liên tục. Nó tạo ra một mô hình tuyến tính dựa trên dữ liệu huấn luyện và sử dụng mô hình này để dự đoán giá trị đầu ra cho các dữ liệu mới.
2. DecisionTreeRegressor là một thuật toán học máy sử dụng cây quyết định để dự đoán giá trị đầu ra liên tục. Nó phân chia tập dữ liệu thành các phân vùng dựa trên các đặc tính và quyết định tại mỗi nút của cây. DecisionTreeRegressor có khả năng xử lý dữ liệu số và phân loại, mô hình hóa mối quan hệ phi tuyến và xử lý dữ liệu thiếu và nhiễu.
3. RandomForestRegressor là một thuật toán học máy được sử dụng để dự đoán giá trị đầu ra liên tục bằng cách kết hợp nhiều cây quyết định (Decision Tree) thành một mô hình. Nó giúp giảm hiện tượng overfitting và có khả năng xử lý dữ liệu thiếu và nhiễu.
4. Lasso là một thuật toán học máy được sử dụng trong bài toán hồi quy để tìm một mô hình tối ưu và thực hiện lựa chọn đặc trưng. Nó áp dụng phạt lên các hệ số của các đặc trưng không quan trọng và giúp giảm độ phức tạp của mô hình.
5. XGBRegressor là một thuật toán học máy sử dụng trong bài toán hồi quy. Nó được xây dựng dựa trên thuật toán Gradient Boosting và sử dụng cây quyết định làm bộ phân loại cơ sở. XGBRegressor có khả năng xử lý các bài toán hồi quy phức tạp và tương đối nhanh chóng. Nó cung cấp các thông số cấu hình linh hoạt cho việc tinh chỉnh mô hình.
6. AdaBoostRegressor là một thuật toán học máy sử dụng trong bài toán hồi quy. Nó tạo ra một chuỗi các mô hình hồi quy yếu và kết hợp chúng để tạo ra một mô hình hồi quy mạnh. AdaBoostRegressor có khả năng xử lý các bài toán hồi quy phức tạp và tránh overfitting.
7. GradientBoostingRegressor là một thuật toán học máy dùng trong bài toán hồi quy. Nó tạo ra một chuỗi các cây quyết định để dự đoán giá trị đầu ra. Thuật toán này cải thiện dự đoán sau mỗi bước bằng cách tìm kiếm các cây quyết định mới để xử lý các sai số còn lại. GradientBoostingRegressor có khả năng xử lý bài toán hồi quy phức tạp và cung cấp kết quả chính xác.
8. CatBoostRegressor là một thuật toán học máy được sử dụng trong bài toán hồi quy. Nó tối ưu hóa đặc biệt cho dữ liệu có đặc tính dạng chuỗi và tự động xử lý các biến category. Nó cung cấp tính năng kiểm soát quá khớp và tăng tốc độ huấn luyện.

## 4.5 XÂY DỰNG THÀNH PHẦN

* Khai báo thư viện

import pandas as pd

import numpy as np

import seaborn as sns

import keras

import matplotlib.pyplot as plt

from datetime import timedelta

import statsmodels.api as sm

from scipy.stats import ttest\_ind

from math import sqrt

import xgboost as xgb

from catboost import CatBoostRegressor

import tensorflow as tf

from tensorflow import keras

from keras.models import Sequential

from keras.layers import Dense, Dropout, Activation

from tensorflow.keras.optimizers import Adam

from sklearn import set\_config

from sklearn.compose import ColumnTransformer

from sklearn.ensemble import ExtraTreesRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.ensemble import AdaBoostRegressor

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.pipeline import Pipeline

from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error

from sklearn.preprocessing import FunctionTransformer

from sklearn.preprocessing import OneHotEncoder

from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import Lasso

from sklearn.tree import DecisionTreeRegressor

from sklearn.model\_selection import train\_test\_split, GridSearchCV, cross\_val\_score, KFold

from sklearn.feature\_selection import mutual\_info\_regression

sns.set\_style("whitegrid")

Khai báo các thư viện cần thiết cho đề tài như:

Pandas, numpy, datetime, statsmodels dùng để xử lý dữ liệu.

Seaborn, matplotlib dùng để vẽ đồ thị.

Keras, tensorflow dùng trong xây dựng và huấn luyện mạng nơ-ron.

Sklearn dùng trong xây dựng các mô hình học máy và đánh giá kết quả.

* Cài đặt cấu hình

# Định dạng ngày tháng (ddmmyyyy)

DATE\_FORMAT = '%d/%m/%Y'

# Gán nhãn cho khoảng thời gian trong ngày ( Departure Time & Arrival Time )

TIME\_PERIOD\_LABELS = {

    1: 'Late Night',

    2: 'Early Morning',

    3: 'Morning',

    4: 'Afternoon',

    5: 'Evening',

    6: 'Night'

}

# Định dạng giờ phút để tính toán thời gian bay (hhmm)

TIME\_FORMAT = '%H:%M'

# Đổi tên các cột

RENAME\_COLUMNS = {

    'airline': 'Airline',

    'from': 'Source City',

    'to': 'Destination City',

    'class': 'Journey Class',

    'price': 'Ticket Price'

}

# Sắp xếp các cột theo thứ tự

COLUMN\_ARRANGEMENT = [

    'Airline', 'Source City', 'Departure Time',

    'Number of Stops', 'Destination City', 'Arrival Time',

    'Trip Duration', 'Date of Journey', 'Days Left',

    'Day of Week', 'Flight Code', 'Journey Class', 'Ticket Price'

]

# Sắp xếp dữ liệu theo tiêu chí

SORT\_DATA = [

    'Days Left',

    'Ticket Price'

]

# Loại bỏ các cột không cần thiết

DROP\_CLEAN\_FLIGHTS = [

    'ch\_code',

    'num\_code',

    'stop',

    'date',

    'price'

]

# Chuyển đổi kiểu dữ liệu của các cột

TRANSFORM\_DATATYPE = [

    'Days Left',

    'Ticket Price',

    'Number of Stops'

]

# Loại bỏ các cột không cần thiết liên quan đến thời gian

DROP\_TIME\_COL = [

    'hour',

    'minute',

    'time\_taken',

    'dep\_time',

    'arr\_time'

]

# Danh sách cho hàm plots visualisation

list1 = ['Source City', 'Destination City', 'Airline', 'Arrival Time', 'Departure Time']

# Màu sắc cho hàm plots visualisation

list1\_colors = ['coral', 'orange', 'green', 'darkseagreen', 'darkslategrey']

# Màu sắc cho biểu đồ đường của các hãng hàng không

airline\_colors = {

    'Vistara': 'coral',

    'Indigo': 'orange',

    'Air India': 'green',

    'SpiceJet': 'brown',

    'AirAsia': 'darkslategrey',

    'GO FIRST': 'blue'

}

#Khai báo parameter grid cho GridSearchCV trong decision tree

param\_grid = {

    'max\_depth': [None, 5, 10],

    'min\_samples\_leaf': [1, 2],

    'min\_samples\_split': [2, 5, 10],

}

#Khai báo parameter grid

param\_grid = {

    'n\_estimators': [50, 100, 200],

    'max\_depth': [10, 20, 30],

    'random\_state': [75]

}

* Xử lý dữ liệu

def concat\_flights(df):

    # Đọc file csv đã tải lên.

    df\_business = pd.read\_csv('/content/sample\_data/business.csv')

    df\_economy = pd.read\_csv('/content/sample\_data/economy.csv')

    # Gán giá trị dựa trên Journey Class

    df\_business['class']='Business'

    df\_economy['class']='Economy'

    # Kết hợp hai tập dữ liệu thành một và lưu chúng vào một dataframe

    df = pd.concat([df\_business,df\_economy], axis =0)

    # Hiển thị các hàng trong dataframe sau khi đã kết hợp

    df.head(8)

    return df

def clean\_flights(df):

    # Thay thế chuỗi bằng số lượng điểm dừng và gán vào cột mới

    df['Number of Stops']= df['stop'].str.slice(start=0, stop=1).replace('n','0')

    # Thay đổi định dạng ngày và gán vào cột mới

    df['Date of Journey']= df['date'].str.replace('-','/')

    # Thay thế dấu ',' từ chuỗi giá và gán vào cột mới

    df['Ticket Price']= df['price'].str.replace(',','')

    # Ghép hai cột và gán vào cột mới

    df['Flight Code'] =  df['ch\_code'] + "-" + df['num\_code'].astype(str)

    # Loại bỏ các cột cũ, 'DROP\_CLEAN\_FLIGHTS' từ tệp cấu hình

    df = df.drop(DROP\_CLEAN\_FLIGHTS, axis = 1)

    return df

def transform\_date(df):

    # Thay đổi kiểu dữ liệu từ chuỗi thành DateTime, định dạng từ 'DATE\_FORMAT' trong tệp cấu hình

    df['Date of Journey'] = pd.to\_datetime(df['Date of Journey'], format=DATE\_FORMAT)

    return df

def transform\_days\_left(df):

    # Tính toán số ngày còn lại từ ngày khởi hành

    df['Days Left'] = (df['Date of Journey'] + timedelta(days=1)) - min(df['Date of Journey'])

    df['Days Left'] = df['Days Left'].astype(str).str.slice(start=0, stop=2)

    # Thay đổi kiểu dữ liệu, "TRANSFORM\_DATATYPE" từ tệp cấu hình

    df[TRANSFORM\_DATATYPE] = df[TRANSFORM\_DATATYPE].astype(int)

    return df

def transform\_day\_of\_week(df):

    # Gán tên ngày vào một cột mới dựa trên Ngày khởi hành

    df['Day of Week'] = df['Date of Journey'].dt.day\_name()

    return df

def transform\_departure\_time(df):

    # Chuyển đổi thời gian khởi hành thành giờ

    dep\_time = (pd.to\_datetime(df['dep\_time'], format=TIME\_FORMAT).dt.hour % 24 + 4) // 4

    dep\_time.replace(TIME\_PERIOD\_LABELS, inplace=True)

    # Gán cột dep\_time vào cột mới

    df['Departure Time'] = dep\_time

    return df

def transform\_arrival\_time(df):

    # Chuyển đổi thời gian đến thành giờ

    arr\_time = (pd.to\_datetime(df['arr\_time'], format=TIME\_FORMAT).dt.hour % 24 + 4) // 4

    arr\_time.replace(TIME\_PERIOD\_LABELS, inplace=True)

    # Gán cột arr\_time vào cột mới

    df['Arrival Time'] = arr\_time

    return df

def transform\_trip\_duration(df):

    # Chuyển đổi chuỗi từ cột time\_taken thành giờ và phút

    hour = df['time\_taken'].str.split('h ', n=1, expand=True)

    df['hour'] = hour[0]

    df['minute'] = hour[1].str.replace('m', '')

    df['minute'] = np.where(df['minute'] == "", 0, df['minute'])

    # Tính toán thời lượng trong các giá trị float để phân tích tốt hơn trong các mô hình

    df['Trip Duration'] = (df['hour'].astype(float) + (df['minute'].astype(float))/60).round(2)

    # Loại bỏ các cột thời gian tạm thời, 'DROP\_TIME\_COL' từ tệp cấu hình

    df = df.drop(DROP\_TIME\_COL, axis=1)

    return df

def transform\_rename\_columns(df):

    # Đổi tên các cột, RENAME\_COLUMNS từ tệp cấu hình

    df.rename(columns= RENAME\_COLUMNS, inplace=True)

    # Sắp xếp lại các cột theo định dạng mong muốn, 'COLUMN\_ARRANGEMENT' từ tệp cấu hình

    df = df[COLUMN\_ARRANGEMENT]

    return df

def transform\_sort(df):

    # Sắp xếp dữ liệu, SORT\_DATA từ tệp cấu hình

    df = df.sort\_values(by= SORT\_DATA)

    return df

def drop\_airlines(df):

    # Loại bỏ một số hàng trong Airline, do số lượng dữ liệu có sẵn ít

    df = df[df['Airline'] != 'StarAir']

    df = df[df['Airline'] != 'Trujet']

    return df

def transform\_flights(df):

    # Gọi tất cả các hàm và gán chúng vào dataframe gốc

    df = transform\_date(df)

    df = transform\_days\_left(df)

    df = transform\_day\_of\_week(df)

    df = transform\_departure\_time(df)

    df = transform\_arrival\_time(df)

    df = transform\_trip\_duration(df)

    df = transform\_rename\_columns(df)

    df = transform\_sort(df)

    df = drop\_airlines(df)

    return df

# Tạo Pipeline để giúp trực quan hóa sự chuyển đổi trong các cột

# Bước đầu tiên xử lý các biến đổi liên quan đến ngày

transform\_day = ColumnTransformer([

    ('transform\_date', FunctionTransformer(transform\_date), ['date\_column']),

    ('transform\_days\_left', FunctionTransformer(transform\_days\_left), ['days\_left\_column']),

    ('transform\_day\_of\_week', FunctionTransformer(transform\_day\_of\_week), ['day\_of\_week\_column'])

])

# Bước thứ hai xử lý các biến đổi liên quan đến thời gian

transform\_time = ColumnTransformer([

    ('transform\_departure\_time', FunctionTransformer(transform\_departure\_time), ['departure\_time\_column']),

    ('transform\_arrival\_time', FunctionTransformer(transform\_arrival\_time), ['arrival\_time\_column']),

    ('transform\_trip\_duration', FunctionTransformer(transform\_trip\_duration), ['trip\_duration\_column'])

])

# Bước thứ ba xử lý các biến đổi liên quan đến cột

transform\_columns = ColumnTransformer([

    ('transform\_rename\_columns', FunctionTransformer(transform\_rename\_columns), None),

    ('transform\_sort', FunctionTransformer(transform\_sort), None)

])

# Tạo Pipeline và gán các bước biến đổi

preprocess\_pipeline = Pipeline([

    ('transform\_day', transform\_day),

    ('transform\_time', transform\_time),

    ('transform\_columns', transform\_columns),

])

# Hiển thị biểu đồ của Pipeline

set\_config(display="diagram")

preprocess\_pipeline

def preprocess\_pipeline(df):

    # Hàm tiền xử lý để gọi tất cả các hàm theo thứ tự

    df = concat\_flights(df)

    df = clean\_flights(df)

    df = transform\_flights(df)

    df = drop\_airlines(df)

    return df

# Hiển thị thời gian thực thi

%%time

df =[]

# Gán giá trị của df sau khi chạy hàm preprocess\_pipeline vào preprocessed\_df

preprocessed\_df = preprocess\_pipeline(df)

# head() - Trả về một tập con của DataFrame gồm các hàng đầu tiên ( mặc định là 5 ).

preprocessed\_df.head()

# Đọc tệp cleaned\_flight\_dataset đã được lưu từ Tiền xử lý

df = pd.read\_csv('cleaned\_flight\_dataset.csv')

# Bỏ các cột không bắt buộc để phân tích

df = df.drop(['Date of Journey', 'Flight Code'], axis=1)

# Tách tập dữ liệu từ Cột cuối cùng

X = df.iloc[:,:-1]

y = df.iloc[:,-1]

# Áp dụng One-Hot Encoding cho các cột phân loại

ct = ColumnTransformer(transformers=[

    ('ohe',OneHotEncoder(drop='first',handle\_unknown='ignore'),

    ['Airline','Source City','Destination City','Journey Class','Departure Time','Arrival Time', 'Day of Week'])

],remainder='passthrough')

# Điều chỉnh và chuyển đổi mô hình với các giá trị được mã hóa

X= ct.fit\_transform(X)

X

# Thay đổi chiều của y để biến đổi vô hướng

y = np.ravel(y).astype('float32').reshape(-1,1)

y = y.reshape(-1,1)

# Tách tập dữ liệu thành Testing và Training

X\_train, X\_test, y\_train, y\_test= train\_test\_split(X, y, test\_size=0.3, random\_state=0)

# Mở rộng dữ liệu trước khi chạy mô hình - StandardScaler

scaler\_x = StandardScaler(with\_mean=False)

X\_train = scaler\_x.fit\_transform(X\_train)

X\_test = scaler\_x.transform(X\_test)

scaler\_y = StandardScaler(with\_mean=False)

y\_train = scaler\_y.fit\_transform(y\_train)

y\_test = scaler\_y.transform(y\_test)

# Mở rộng dữ liệu trước khi chạy mô hình - MaxAbsScaler

minmax\_x = MaxAbsScaler()

X\_train\_m = minmax\_x.fit\_transform(X\_train)

X\_test\_m = minmax\_x.transform(X\_test)

minmax\_y = MaxAbsScaler()

y\_train\_m = minmax\_y.fit\_transform(y\_train)

y\_test\_m = minmax\_y.transform(y\_test)

# Thực hiện lại test\_train\_split để có dữ liệu mới

X\_train, X\_test, y\_train, y\_test= train\_test\_split(X, y, test\_size=0.3, random\_state=36)

# Chuyển đổi y thành float để prediction

y = np.array(y).astype('float32')

# Chọn các hàng có hạng Hành trình là Thương gia hoặc Phổ thông

business\_flights = df[df['Journey Class'] == 'Business']

economy\_flights = df[df['Journey Class'] == 'Economy']

# Tạo dataframe rỗng với các cột city, p\_class, mean\_ticket\_price

result\_df = pd.DataFrame(columns=['city', 'p\_class', 'mean\_ticket\_price'])

# Xử lý dữ liệu gốc kết quả cuối cùng trả về result\_df chứa các thông tin về thành phố, class, giá trị trung bình của giá vé

# Vòng lặp đầu tiên để duyệt qua từng thành phố trong cột 'Source City'

for city in df['Source City'].unique():

    for p\_class in df['Journey Class'].unique():

        # Tính giá trị trung bình của cột Ticket Price.

        mean\_ticket\_price = df['Ticket Price'].loc[(df['Source City'] == city) & (df['Journey Class'] == p\_class)].mean()

        # Thêm hàng mới vào result\_df với các giá trị city, p\_class và mean\_ticket\_price tương ứng

        result\_df = result\_df.append({'city': city, 'p\_class': p\_class, 'mean\_ticket\_price': mean\_ticket\_price},ignore\_index=True)

# Tạo dataframe rỗng với các cột city, p\_class, mean\_ticket\_price

result\_df = pd.DataFrame(columns=['city', 'p\_class', 'mean\_ticket\_price'])

# Xử lý dữ liệu gốc kết quả cuối cùng trả về result\_df chứa các thông tin về thành phố, class, giá trị trung bình của giá vé

# Vòng lặp đầu tiên để duyệt qua từng thành phố trong cột 'Destination City'

for city in df['Destination City'].unique():

    for p\_class in df['Journey Class'].unique():

        # Tính giá trị trung bình của cột Ticket Price.

        mean\_ticket\_price = df['Ticket Price'].loc[(df['Destination City'] == city) & (df['Journey Class'] == p\_class)].mean()

        # Thêm hàng mới vào result\_df với các giá trị city, p\_class và mean\_ticket\_price tương ứng

        result\_df = result\_df.append({'city': city, 'p\_class': p\_class, 'mean\_ticket\_price': mean\_ticket\_price},

                                     ignore\_index=True)

# Thêm cột Route vào df ( kết hợp Source City và Destination City )

df["Route"] = df['Source City'] + '-' + df['Destination City']

# Tính toán ma trận tương quan giữa các cặp biến trong DataFrame

corr = df.corr()

# Nhóm dữ liệu theo cột Route, tính giá trị trung bình của giá vé và sắp xếp theo thứ tự giảm dần theo Economy Class

e\_route\_df = df.groupby('Route').apply(lambda df: df['Ticket Price'].loc[df['Journey Class'] == 'Economy'].mean()).sort\_values(ascending=False)

# Nhóm dữ liệu theo cột Route, tính giá trị trung bình của giá vé và sắp xếp theo thứ tự giảm dần theo Business Class

b\_route\_df = df.groupby('Route').apply(lambda df: df['Ticket Price'].loc[df['Journey Class'] == 'Business'].mean()).sort\_values(ascending=False)

# Tạo dataframe class\_df được nhóm theo cột Journey Class và giá trị trung bình của giá vé

class\_df = df.groupby('Journey Class').apply(lambda df: df['Ticket Price'].mean())

# Chuyển đổi chuỗi kết quả thành df và đặt lại index

class\_df = class\_df.to\_frame().reset\_index()

class\_df.dtypes

# Chuẩn bị dữ liệu

df\_temp = df[['Departure Time', 'Arrival Time', 'Ticket Price']]

# Tạo các khoảng (bins) cho cột "Trip Duration" của DataFrame

bins = np.linspace(start=df['Trip Duration'].min(), stop=df['Trip Duration'].max(), num=4)

# Gán nhãn cho các khoảng

labels = ['short', 'moderate', 'long']

# Gán nhãn cho cột "Trip Duration" bằng các khoảng đã tạo trước đó

df['Trip Duration'] = pd.cut(df['Trip Duration'], bins=bins, labels=labels, include\_lowest=True)

# Đếm số lượng giá trị trong mỗi nhãn

df['Trip Duration'].value\_counts()

# Xử lý dữ liệu tạo ra các df Mean/ Max/ Min Price

mean\_duration = df.groupby(['Trip Duration', 'Journey Class'])['Ticket Price'].mean().to\_frame()

max\_duration = df.groupby(['Trip Duration', 'Journey Class'])['Ticket Price'].max().to\_frame()

min\_duration = df.groupby(['Trip Duration', 'Journey Class'])['Ticket Price'].max().to\_frame()

# Xử lý dữ liệu tạo ra các df Mean/ Max/ Min Price

mean\_stops = df.groupby(['Number of Stops', 'Journey Class'])['Ticket Price'].mean().to\_frame()

max\_stops = df.groupby(['Number of Stops', 'Journey Class'])['Ticket Price'].max().to\_frame()

min\_stops = df.groupby(['Number of Stops', 'Journey Class'])['Ticket Price'].max().to\_frame()

def preprocessing(df):

    # Mã hóa biến thứ tự "stops" và "class".

    df["Number of Stops"] = df["Number of Stops"].replace({'zero':0,'one':1,'two\_or\_more':2}).astype(int)

    df["Journey Class"] = df["Journey Class"].replace({'Economy':0,'Business':1}).astype(int)

    # Tạo các biến giả (dummy variables) cho các thành phố, thời gian khởi hành và hãng hàng không.

    dummies\_variables = ["Airline","Source City","Destination City","Departure Time","Arrival Time"]

    dummies = pd.get\_dummies(df[dummies\_variables], drop\_first= True)

    df = pd.concat([df,dummies],axis=1)

    # Xóa các cột gốc của thành phố, thời gian khởi hành và hãng hàng không.

    df = df.drop(["Airline","Source City","Destination City","Departure Time","Arrival Time"],axis=1)

    return df

#  Tạo một ma trận tam giác trên (upper triangular matrix) có kích thước giống với ma trận tương quan của DataFrame

mask = np.triu(np.ones\_like(df\_preprocessed.corr(), dtype=bool))

def make\_mi\_scores(X, y):

    # Tạo một bản sao của df X và mã hóa dữ liệu thành kiểu nguyên

    X = X.copy()

    for colname in X.select\_dtypes(["object", "category"]):

        X[colname], \_ = X[colname].factorize()

    # Kiểm tra kiểu dữ liệu của tất cả các discrete features

    discrete\_features = [pd.api.types.is\_integer\_dtype(t) for t in X.dtypes]

    # Tính điểm thông tin chung giữa biến đầu vào và biến mục tiêu

    mi\_scores = mutual\_info\_regression(X, y, discrete\_features=discrete\_features, random\_state=0)

    # Lưu kết quả vào một Series pandas được sắp xếp giảm dần

    mi\_scores = pd.Series(mi\_scores, name="MI Scores", index=X.columns)

    mi\_scores = mi\_scores.sort\_values(ascending=False)

    return mi\_scores

# Loại bỏ cột "Date of Journey" khỏi DataFrame ( do mi\_scores không hỗ trợ kiểu dữ liệu datetime64)

df\_preprocessed = df\_preprocessed.drop("Date of Journey", axis=1)

# Tạo một bản sao X của df\_preprocessed và tách cột "Ticket Price" ra khỏi X

X = df\_preprocessed.copy()

y = X.pop("Ticket Price")

# Tính toán điểm thông tin chung và in danh sách điểm thông tin chung theo thứ tự giảm dần

mi\_scores = make\_mi\_scores(X, y)

print(mi\_scores.sort\_values(ascending=False))

* Xây dựng model

# Xác định Model Deep Learning

ANN\_model = keras.Sequential()

ANN\_model.add(Dense(100, input\_dim = 35))

ANN\_model.add(Activation('relu'))

ANN\_model.add(Dense(150))

ANN\_model.add(Activation('relu'))

ANN\_model.add(Dropout(0.25))

ANN\_model.add(Dense(150))

ANN\_model.add(Activation('relu'))

ANN\_model.add(Dense(150))

ANN\_model.add(Activation('linear'))

ANN\_model.add(Dense(1))

ANN\_model.compile(loss = 'mse', optimizer = 'adam')

ANN\_model.summary()

# Gán model cho các biến

lr = LinearRegression()

dt1 = DecisionTreeRegressor()

dt2 = DecisionTreeRegressor()

rf = RandomForestRegressor()

ls = Lasso()

XGB = xgb.XGBRegressor()

abr = AdaBoostRegressor()

gb = GradientBoostingRegressor()

cat = CatBoostRegressor()

# Tạo một pipeline chứa danh sách các mô hình hoặc bước xử lý dữ liệu.

pipeline= [lr, dt1, dt2, rf, ls, XGB, abr, gb, cat]

* Training và Fitting model

# Biên dịch mô hình ANN theo Standard Scalar

ANN\_model.compile(optimizer = 'Adam', loss = 'mean\_squared\_error')

# Training model ANN

epochs\_hist = ANN\_model.fit(X\_train, y\_train, epochs = 20, batch\_size = 20)

# Đánh giá hiệu suất của model

result = ANN\_model.evaluate(X\_test, y\_test)

# Độ chính xác của model

accuracy\_ANN = 1 - result

# Biên dịch mô hình ANN theo MaxAbsScaler

epochs\_hist\_m = ANN\_model.fit(X\_train\_m, y\_train\_m, epochs = 20, batch\_size = 20)

# Đánh giá hiệu suất của model

result\_m = ANN\_model.evaluate(X\_test\_m, y\_test\_m)

# Độ chính xác của model

accuracy\_ANN\_m = 1 - result\_m

print(f"Accuracy: {round((accuracy\_ANN\_m)\*100, 3)}%")

#So sánh độ chính xác từ mô hình Standard Scaler và mô hình MinMaxScaler

print(f"Accuracy (StandardScaler) : {round((accuracy\_ANN)\*100, 3)}%")

print(f"Accuracy (MinMaxScaler)   : {round((accuracy\_ANN\_m)\*100, 3)}%")

# Dự đoán mô hình

y\_predict\_m = ANN\_model.predict(X\_test\_m)

# Fitting mô hình Linear Regression

lr.fit(X\_train, y\_train)

# Kết quả dự đoán

y\_pred\_lr = lr.predict(X\_test)

y\_pred\_lr

# Sử dụng cross validation và grid search Decision Tree

cv = KFold(n\_splits=10, random\_state=42, shuffle=True)

dt1 = GridSearchCV(estimator=dt1, param\_grid=param\_grid, cv=cv, scoring='neg\_mean\_squared\_error')

# Fitting

dt1.fit(X\_train,y\_train)

# In các tham số tốt nhất của Decision Tree dựa trên các tham số được cung cấp

print(f"Best hyperparameters: {dt1.best\_params\_}")

print(f"Best cross-validation score: {-dt1.best\_score\_:.2f}")

# Điều chỉnh mô hình

dt1 = DecisionTreeRegressor(max\_depth=None, min\_samples\_split=10, min\_samples\_leaf= 2)

# Fitting

dt1.fit(X\_train, y\_train)

# Kết quả dự đoán

y\_pred\_dt = dt1.predict(X\_test)

y\_pred\_dt

# Mô hình với max\_depth = 5

dt2 = DecisionTreeRegressor(max\_depth=5, min\_samples\_split=10)

# Fitting

dt2.fit(X\_train, y\_train)

# Kết quả dự đoán

y\_pred = dt2.predict(X\_test)

y\_pred

#Mô hình Random forest với hyperparameters được điều chỉnh

rf = RandomForestRegressor(n\_estimators=100, max\_depth=20, random\_state=75)

# Fitting

rf.fit(X\_train, y\_train)

# Kết quả dự đoán

y\_pred\_rf = rf.predict(X\_test)

y\_pred\_rf

# Mô hình Lasso

ls = Lasso(alpha = 50,max\_iter = 100,tol=0.1)

# Fitting

ls.fit(X\_train, y\_train)

# Kết quả dự đoán

y\_pred\_ls = ls.predict(X\_test)

y\_pred\_ls

# Mô hình XGBRegressor

XGB = xgb.XGBRegressor(objective ='reg:squarederror', n\_estimators = 10, seed = 42)

# Fitting

XGB.fit(X\_train, y\_train)

# Kết quả dự đoán

y\_pred\_xgb = XGB.predict(X\_test)

y\_pred\_xgb

# Mô hình AdaBoostRegressor

abr = AdaBoostRegressor(n\_estimators=50, learning\_rate=0.1, random\_state=42)

# Fitting

abr.fit(X\_train, y\_train)

# Kết quả dự đoán

y\_pred\_abr = abr.predict(X\_test)

y\_pred\_abr

# Mô hình GradientBoostingRegressor

gb = GradientBoostingRegressor(loss='squared\_error', learning\_rate=0.1, n\_estimators=100, max\_depth=3)

# Fitting

gb.fit(X\_train, y\_train)

# Kết quả dự đoán

y\_pred\_gb = gb.predict(X\_test)

y\_pred\_gb

# Mô hình CatBoostRegressor

cat = CatBoostRegressor(loss\_function='RMSE', learning\_rate=0.05, n\_estimators=200, max\_depth=6)

# Fitting

cat.fit(X\_train, y\_train)

# Kết quả dự đoán

y\_pred\_cat = cat.predict(X\_test)

y\_pred\_cat

* Vẽ đồ thị

# Vẽ tất cả các biểu đồ trên cùng một dòng

fig, axes = plt.subplots(1, len(list1), figsize=(20, 5))

# Điều chỉnh khoảng cách giữa các biểu đồ

fig.subplots\_adjust(wspace=0.4)

# Lặp để vẽ biểu đồ cho mỗi cột, list1 từ tệp cấu hình

for i, l in enumerate(list1):

    counts = df[[l]].value\_counts().reset\_index(name='count')

    counts.plot(x=l, y='count', kind='bar', ax=axes[i], color=list1\_colors[i % len(list1\_colors)])

    axes[i].set\_title(l)

    axes[i].set\_xlabel(l)

    axes[i].set\_ylabel('Count')

# Hiển thị biểu đồ

plt.show()

# Vẽ biểu đồ histogram sử dụng seaborn với KDE

sns.histplot(data=df, x='Ticket Price', color="green", label="Airline", kde=True)

sns.barplot(data=df, x='Airline', y='Ticket Price', hue='Journey Class')

sns.lineplot(data=df, x='Days Left', y='Ticket Price', hue='Journey Class', size='Airline')

grid = sns.FacetGrid(economy\_flights, col="Journey Class", height=6)

grid.map(sns.lineplot, 'Departure Time', 'Ticket Price', color='blue', label='Departure Time')

grid.map(sns.lineplot, 'Arrival Time', 'Ticket Price', color='red', label='Arrival Time')

grid.set\_axis\_labels("Departure Times", "Ticket Price")

grid.add\_legend()

grid.set\_titles("Price Trend for Economy Class")

# Vẽ biểu đồ Scatterplot đa biến hiển thị giá trị trung bình của giá vé theo thành phố và lớp hành trình

sns.set\_style('whitegrid')

g = sns.relplot(data=result\_df, x='city', y='mean\_ticket\_price', hue='p\_class')

for \_, row in result\_df.iterrows():

    g.ax.text(row['city'], row['mean\_ticket\_price'], f'{row["mean\_ticket\_price"]:.2f}',

              color='black', ha='center', va='bottom', fontsize=8)

# Chú thích biểu đồ

plt.xticks(rotation=45, ha='right')

plt.title('Mean Ticket Prices by City and Passenger Class - Source City')

plt.xlabel('City')

plt.ylabel('Mean Ticket Price')

plt.show()

# Vẽ biểu đồ Heatmap đa biến hiển thị giá trị trung bình của giá vé theo thành phố và class

sns.set\_style('whitegrid')

g = sns.heatmap(data=result\_df.pivot('city', 'p\_class', 'mean\_ticket\_price'), cmap='RdYlBu\_r', annot=True, fmt='.2f')

g.set\_title('Mean Ticket Prices by City and Passenger Class - Source City')

g.set\_xlabel('Passenger Class')

g.set\_ylabel('City')

plt.show()

# Vẽ biểu đồ Scatterplot đa biến hiển thị giá trị trung bình của giá vé theo thành phố và lớp hành trình

sns.set\_style('whitegrid')

g = sns.relplot(data=result\_df, x='city', y='mean\_ticket\_price', hue='p\_class')

for \_, row in result\_df.iterrows():

    g.ax.text(row['city'], row['mean\_ticket\_price'], f'{row["mean\_ticket\_price"]:.2f}',

              color='black', ha='center', va='bottom', fontsize=8)

# Chú thích biểu đồ

plt.xticks(rotation=45, ha='right')

plt.title('Mean Ticket Prices by City and Passenger Class - Destination City')

plt.xlabel('City')

plt.ylabel('Mean Ticket Price')

plt.show()

# Vẽ biểu đồ Heatmap đa biến hiển thị giá trị trung bình của giá vé theo thành phố và class

sns.set\_style('whitegrid')

g = sns.heatmap(data=result\_df.pivot('city', 'p\_class', 'mean\_ticket\_price'), cmap='RdYlBu\_r', annot=True, fmt='.2f')

g.set\_title('Mean Ticket Prices by City and Passenger Class - Destination City')

g.set\_xlabel('Passenger Class')

g.set\_ylabel('City')

plt.show()

# Vẽ biểu đồ scatter plot sử dụng dữ liệu từ "e\_route\_df"

# Thể hiện giá trị trung bình của vé theo từng tuyến đường trong lớp hạng "Economy"

sns.set\_style('whitegrid')

g = sns.scatterplot(data=e\_route\_df, y='Route', x=e\_route\_df.values)

g.set\_title("Route-wise mean prices of Economy Class")

g.set\_xlabel("Ticket Price")

# Vẽ biểu đồ scatter plot sử dụng dữ liệu từ "b\_route\_df"

# Thể hiện giá trị trung bình của vé theo từng tuyến đường trong lớp hạng "Business"

sns.set\_style('whitegrid')

g = sns.scatterplot(data=b\_route\_df, y='Route', x=b\_route\_df.values)

g.set\_title("Route-wise mean prices of Business Class")

g.set\_xlabel("Ticket Price")

# Vẽ biểu đồ cột từ dữ liệu của class\_df theo Journey Class

sns.barplot(data=class\_df, x='Journey Class', y=0)

plt.xlabel('Ticket Price')

plt.title('Comparison of Business and Economy Ticket Prices')

plt.ylabel("Ticket Price")

plt.show()

# Vẽ đồ thị Flight Prices dựa trên Departure Time

plot\_flight\_prices(df, 'Departure Time', 'Ticket Price', 'Airline',

                   airline\_colors, 'Flight Prices based on Different Departure Time', 'H')

def plot\_flight\_prices(data, x\_col, y\_col, hue\_col, palette, title, layout):

    # Sắp xếp DataFrame chứa các chuyến bay Business theo thứ tự thời gian khởi hành

    business\_flights\_sorted = business\_flights.sort\_values(x\_col)

    # Sắp xếp DataFrame chứa các chuyến bay Economy theo thứ tự thời gian khởi hành

    economy\_flights\_sorted = economy\_flights.sort\_values(x\_col)

    # Thiết lập figure với hai subplot

    if (layout =='H'):

        fig, axs = plt.subplots(1, 2, figsize=(20, 7))

    else:

        fig, axs = plt.subplots(2, 1, figsize=(15, 15))

    # Vẽ biểu đồ phân phối giá vé theo các thời gian khởi hành khác nhau cho chuyến bay Business

    sns.lineplot(data=business\_flights\_sorted, x=x\_col, y=y\_col, hue=hue\_col, ax=axs[0], palette=palette)

    axs[0].set\_title('Business Flights', fontsize =14)

    axs[0].get\_legend().remove()

    axs[0].grid(color='lightgrey', linestyle='--', linewidth=0.9)

    # Vẽ biểu đồ phân phối giá vé theo các thời gian khởi hành khác nhau cho chuyến bay Economy

    sns.lineplot(data=economy\_flights\_sorted, x=x\_col, y=y\_col, hue=hue\_col, ax=axs[1], palette=palette)

    axs[1].set\_title('Economy Flights', fontsize =14)

    axs[1].get\_legend().remove()

    axs[1].grid(color='lightgrey', linestyle='--', linewidth=0.9)

    # Vẽ legend chung cho cả hai biểu đồ ở vị trí right center

    handles, labels = axs[1].get\_legend\_handles\_labels()

    unique\_labels = list(set(labels))

    unique\_handles = [handles[labels.index(label)] for label in unique\_labels]

    fig.legend(unique\_handles, unique\_labels,loc='center right', bbox\_to\_anchor=(1.0, 0.5), fancybox= True)

    # Đặt tiêu đề cho figure

    fig.suptitle(title, fontsize =18)

    plt.grid(color = 'lightgrey', linestyle = '--', linewidth = 0.9)

    plt.show()

# Vẽ đồ thị Flight Prices dựa trên Source City

plot\_flight\_prices(df, 'Source City', 'Ticket Price', 'Airline',

                   airline\_colors, 'Flight Prices based on Different Source City', 'H')

# Vẽ đồ thị Flight Prices dựa trên Destination City

plot\_flight\_prices(df, 'Destination City', 'Ticket Price', 'Airline',

                   airline\_colors, 'Flight Prices based on Different Destination City', 'H')

# Vẽ đồ thị Flight Prices dựa trên Number of Stops

plot\_flight\_prices(df, 'Number of Stops', 'Ticket Price', 'Airline',

                   airline\_colors, 'Flight Prices based on Number of Stops', 'H')

# Vẽ đồ thị Flight Prices dựa trên Trip Duration

plot\_flight\_prices(df, 'Trip Duration', 'Ticket Price', 'Airline',

                   airline\_colors, 'Flight Prices based on Trip Duration', 'V')

# Vẽ đồ thị Flight Prices dựa trên Trip Duration

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

sns.lineplot(data=df, x='Trip Duration', y='Ticket Price', hue='Journey Class')

plt.xticks(range(0, 52, 5))

plt.show()

# Vẽ đồ thị Flight Prices dựa trên Day of Week

plot\_flight\_prices(df, 'Day of Week', 'Ticket Price', 'Airline',

                   airline\_colors, 'Flight Prices based on Day of Week', 'H')

# Vẽ biểu đồ Violin cho sự thay đổi của Flight Price dựa trên Number of Stops

fig, axs = plt.subplots(1, 2, figsize=(20, 7))

# Biểu đồ cho chuyến bay Business

sns.violinplot(x='Number of Stops', y='Ticket Price', ax=axs[0], data=business\_flights)

axs[0].set\_title('Business Flights', fontsize =14)

# Biểu đồ cho chuyến bay Economy

sns.violinplot(x='Number of Stops', y='Ticket Price', ax=axs[1], data=economy\_flights)

axs[1].set\_title('Economy Flights', fontsize =14)

fig.suptitle('Flight Price Variation based on Stops', fontsize =18)

plt.show()

# Vẽ đồ thị Flight Prices dựa trên Days left

fig = plt.figure(figsize=(10,4))

sns.lineplot(data=df, x='Days Left', y='Ticket Price', hue='Journey Class')

plt.xticks(range(0, 52, 5))

# Vùng có màu nền biểu diễn sự biến đổi giá vé chuyến bay Economy

plt.axhspan(5000, 10000, 0.305, 0.37, alpha=0.2, color='dodgerblue')

# Thêm nhãn cho vùng có màu nền

plt.annotate('Economy Class Price Change', xy=(18, 10000), xytext=(27, 25000),

             arrowprops=dict(facecolor='black', arrowstyle='->'))

plt.annotate('Business Class Price Change', xy=(11, 55000), xytext=(27, 60000),

             arrowprops=dict(facecolor='black', arrowstyle='->'))

# Vùng có màu nền biểu diễn sự biến đổi giá vé chuyến bay Business

plt.axhspan(50000, 55000, 0.175, 0.24, alpha=0.2, color='orange')

plt.show()

# Trực quan hóa

fig1 = sns.boxplot(x='Departure Time', y='Ticket Price', data=df)

plt.title('Ticket price based on Departure Time')

plt.show()

fig2 = sns.boxplot(x='Arrival Time', y='Ticket Price', data=df)

plt.title('Ticket price based on Arrival Time')

plt.show()

# Phân tích thống kê

night\_prices = df\_temp[df\_temp['Departure Time'] == 'Night']['Ticket Price']

early\_morning\_prices = df\_temp[df\_temp['Arrival Time'] == 'Early Morning']['Ticket Price']

t\_statistic, p\_value = ttest\_ind(night\_prices, early\_morning\_prices)

print('T-Statistic: {:.2f}, P-Value: {:.4f}'.format(t\_statistic, p\_value))

fig, axs = plt.subplots (1, 2, gridspec\_kw={'width\_ratios': [5, 3]}, figsize=(25, 5))

# Vẽ đồ thị cột thể hiện sự ảnh hướng của số điểm dừng của từng hãng hàng không đến giá vé Phổ thông ( Economy )

sns.barplot(y = "Ticket Price", x = "Airline",hue="Number of Stops",data = df.loc[df["Journey Class"]=='Economy'].sort\_values("Ticket Price", ascending = False), ax=axs[0])

axs[0].set\_title("Airline prices based on the number of stops  for economy",fontsize=20)

# Vẽ đồ thị cột thể hiện sự ảnh hướng của số điểm dừng của từng hãng hàng không đến giá vé Thương gia ( Business )

sns.barplot(y = "Ticket Price", x = "Airline",hue="Number of Stops",data = df.loc[df["Journey Class"]=='Business'].sort\_values("Ticket Price", ascending = False), ax=axs[1])

axs[1].set\_title("Airline prices based on the number of stops  for business",fontsize=20)

# Vẽ biểu đồ cột hiển thị giá trị trung bình, giá trị lớn nhất và giá trị nhỏ nhất của cột "Ticket Price" dựa trên các nhóm "Trip Duration" và "Journey Class".

fig, axs = plt.subplots(1, 3, figsize=(10, 4))

mean\_duration.unstack().plot(kind='bar', ax=axs[0], title='Mean Price')

max\_duration.unstack().plot(kind='bar', ax=axs[1], title='Max Price')

min\_duration.unstack().plot(kind='bar', ax=axs[2], title='Min Price')

plt.suptitle('Price by Duration and Class')

plt.show()

# Vẽ biểu đồ cột hiển thị giá trị trung bình, giá trị lớn nhất và giá trị nhỏ nhất của cột "Ticket Price" dựa trên nhóm "Number of Stops"

fig, axs = plt.subplots(1, 3, figsize=(10, 4))

mean\_stops.unstack().plot(kind='bar', ax=axs[0], title='Mean Price')

max\_stops.unstack().plot(kind='bar', ax=axs[1], title='Max Price')

min\_stops.unstack().plot(kind='bar', ax=axs[2], title='Min Price')

plt.suptitle('Price by number of stops and Class')

plt.show()

# Trực quan hóa ma trận tương quan bằng bản đồ heatmap

fig = sns.heatmap(corr, annot=True, cmap='coolwarm').get\_figure()

plt.title('Correlation matrix of Flight Data')

plt.show()

# Vẽ biểu đồ heatmap dựa trên ma trận tương quan của DataFrame

plt.figure(figsize = (20,18))

sns.heatmap(data = df\_preprocessed.corr(), mask=mask,annot = True, vmin= -1.0, vmax= 1.0, center = 0, cmap = 'RdBu\_r')

# Tạo figure và axes objects cho subplots

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

# Trực quan hóa dữ liệu trên subplot đầu tiên

ax1.plot(epochs\_hist.history['loss'])

ax1.set\_title('Standard Scalar Training', fontsize=14)

ax1.set\_xlabel('Epoch')

ax1.set\_ylabel('Training & Valid Loss')

ax1.set\_xlim(0,20)

ax1.set\_ylim(0,0.06)

ax1.legend(['Training Loss','Valid Loss'])

# Trực quan hóa dữ liệu trên subplot thứ hai

ax2.plot(epochs\_hist\_m.history['loss'])

ax2.set\_title('MinMax Scalar Training', fontsize=14)

ax2.set\_xlabel('Epoch')

ax2.set\_ylabel('Training & Valid Loss')

ax2.set\_xlim(0,20)

ax2.set\_ylim(0,0.06)

ax2.legend(['Training Loss','Valid Loss'])

# Hiển thị biểu đồ

plt.suptitle('Loss Progress During Different Training', fontsize=18)

plt.legend(['Training Loss','Valid Loss'])

plt.show()

# Trực quan hóa kết quả

plt.plot(figsize=(15, 8))

plt.plot(y\_test\_m, y\_predict\_m, "o", color = 'xkcd:sky blue', alpha = 0.1)

x = np.linspace(\*plt.xlim())

plt.plot(x, x, color='red', linestyle='--')

plt.xlabel('Model Predictions')

plt.ylabel('True Values')

plt.title('True Value v/s Predicted Value (Scaled)')

fig, ((ax1, ax2, ax3), (ax4, ax5, ax6), (ax7, ax8, ax9)) = plt.subplots(3, 3, figsize=(20, 18))

# Mô hình Linear Regression

ax1.scatter(y\_test, y\_pred\_lr, s=2)

ax1.set\_title('Linear Regression Results')

ax1.set\_xlabel('True Values')

ax1.set\_ylabel('Predicted Values')

ax1.set\_ylim(0,125000)

ax1.set\_xlim(0,125000)

# Mô hình Decision Tree Regression

ax2.scatter(y\_test, y\_pred\_dt, s=2)

ax2.set\_title('Decision Tree Regression Results')

ax2.set\_xlabel('True Values')

ax2.set\_ylabel('Predicted Values')

ax2.set\_ylim(0,125000)

ax2.set\_xlim(0,125000)

# Mô hình Random Forest Regression

ax3.scatter(y\_test, y\_pred\_rf, s=2)

ax3.set\_title('Random Forest Regression Results')

ax3.set\_xlabel('True Values')

ax3.set\_ylabel('Predicted Values')

ax3.set\_ylim(0,125000)

ax3.set\_xlim(0,125000)

# Mô hình Decision Tree - Basic Tuned

ax4.scatter(y\_test, y\_pred, s=2)

ax4.set\_title('Decision Tree - Basic Tuned Results')

ax4.set\_xlabel('True Values')

ax4.set\_ylabel('Predicted Values')

ax4.set\_ylim(0,125000)

ax4.set\_xlim(0,125000)

# Mô hình Lasso Regression

ax5.scatter(y\_test, y\_pred\_ls, s=2)

ax5.set\_title('Lasso Regression Results')

ax5.set\_xlabel('True Values')

ax5.set\_ylabel('Predicted Values')

ax5.set\_ylim(0,125000)

ax5.set\_xlim(0,125000)

# Mô hình XGBoost

ax6.scatter(y\_test, y\_pred\_xgb, s=2)

ax6.set\_title('XGBoost Results')

ax6.set\_xlabel('True Values')

ax6.set\_ylabel('Predicted Values')

ax6.set\_ylim(0,125000)

ax6.set\_xlim(0,125000)

# Mô hình AdaBoost

ax7.scatter(y\_test, y\_pred\_abr, s=2)

ax7.set\_title('AdaBoost Results')

ax7.set\_xlabel('True Values')

ax7.set\_ylabel('Predicted Values')

ax7.set\_ylim(0,125000)

ax7.set\_xlim(0,125000)

# Mô hình Gradient Boost

ax8.scatter(y\_test, y\_pred\_gb, s=2)

ax8.set\_title('Gradient Boost Results')

ax8.set\_xlabel('True Values')

ax8.set\_ylabel('Predicted Values')

ax8.set\_ylim(0,125000)

ax8.set\_xlim(0,125000)

# Mô hình CatBoost Regressor

ax9.scatter(y\_test, y\_pred\_cat, s=2)

ax9.set\_title('CatBoost Regressor Results')

ax9.set\_xlabel('True Values')

ax9.set\_ylabel('Predicted Values')

ax9.set\_ylim(0,125000)

ax9.set\_xlim(0,125000)

# Vẽ đường thẳng y = x trên tất cả các subplot

for ax in (ax1, ax2, ax3, ax4, ax5, ax6, ax7, ax8, ax9):

    x = np.linspace(\*plt.xlim())

    ax.plot(x, x, transform=ax.transAxes, ls='--', c='red')

# Điều chỉnh khoảng cách giữa các subplot

fig.subplots\_adjust(wspace=0.4)

plt.suptitle("Comparison Scatter Plots of Different Regression Results", fontsize=20)

plt.show()

# Đồ thị Phân phối tần số Residual error cho tất cả các mô hình

fig, ((ax1, ax2, ax3), (ax4, ax5, ax6), (ax7, ax8, ax9)) = plt.subplots(3, 3, figsize=(20, 18))

# Mô hình Linear Regression

sns.histplot(residual\_lr, kde=True, ax=ax1)

ax1.set\_xlabel('Residual error')

ax1.set\_title('Linear Regression Results')

# Mô hình Decision Tree Regression

sns.histplot(residual\_dt, kde=True, ax=ax2)

ax2.set\_xlabel('Residual error')

ax2.set\_title('Decision Tree Regression Results')

# Mô hình Random Forest Regression

sns.histplot(residual\_rf, kde=True, ax=ax3)

ax3.set\_xlabel('Residual error')

ax3.set\_title('Random Forest Regression Results')

# Mô hình Decision Tree - Basic Tuned

sns.histplot(residual, kde=True, ax=ax4)

ax4.set\_xlabel('Residual error')

ax4.set\_title('Decision Tree - Basic Tuned Results')

# Mô hình Lasso Regression

sns.histplot(residual\_ls, kde=True, ax=ax5)

ax5.set\_xlabel('Residual error')

ax5.set\_title('Lasso Regression Results')

# Mô hình XGBoost

sns.histplot(residual\_xgb, kde=True, ax=ax6)

ax6.set\_xlabel('Residual error')

ax6.set\_title('XGBoost Results')

# Mô hình AdaBoost

sns.histplot(residual\_abr, kde=True, ax=ax7)

ax7.set\_xlabel('Residual error')

ax7.set\_title('AdaBoost Results')

# Mô hình Gradient Boost

sns.histplot(residual\_gb, kde=True, ax=ax8)

ax8.set\_xlabel('Residual error')

ax8.set\_title('Gradient Boost Results')

# Mô hình CatBoost Regressor

sns.histplot(residual\_cat, kde=True, ax=ax9)

ax9.set\_xlabel('Residual error')

ax9.set\_title('CatBoost Regressor Results')

# Điều chỉnh khoảng cách giữa các subplot

fig.subplots\_adjust(wspace=0.4)

plt.suptitle('Frequency distribution of residual error', fontsize=20)

plt.show()

* Kết quả mô hình

# Đảo ngược dữ liệu được chia tỷ lệ

y\_predict\_orig\_m = minmax\_y.inverse\_transform(y\_predict\_m)

y\_test\_orig\_m = minmax\_y.inverse\_transform(y\_test\_m)

# Kết quả của mô hình Deep Learning

# Độ chính xác (%)

score = round((accuracy\_ANN\_m)\*100, 3)

# Sai số bình phương trung bình

mse = round(mean\_squared\_error(y\_test\_orig\_m, y\_predict\_orig\_m), 3)

# Sai số bình phương trung bình căn

rmse = round(np.sqrt(mean\_squared\_error(y\_test\_orig\_m, y\_predict\_orig\_m)), 3)

# Hệ số xác định R

r2 = round(r2\_score(y\_test\_orig\_m, y\_predict\_orig\_m), 3)

# Sai số tuyệt đối trung bình

mae = mean\_absolute\_error(y\_test\_orig\_m, y\_predict\_orig\_m)

# Lưu kết quả vào DataFrame

result = pd.DataFrame({

        'Model Name': 'Artificial Neural Network',

        'Accuracy(%)': [score],

        'Mean Squared Error (MSE)': [mse],

        'Root Mean Squared Error (RMSE)': [rmse],

        'R-squared (R2-Score)': [r2],

        'Mean Absolute Error (MAE)': [mae]

    })

# In kết quả

result

# Hàm tính toán hiệu suất của pipeline

def evaluate\_pipeline(pipeline, y\_pred\_all, model\_name):

    # Độ chính xác (%)

    score = round((pipeline.score(X\_test, y\_test)\*100), 2)

    # Sai số bình phương trung bình

    mse = round(mean\_squared\_error(y\_test, y\_pred\_all), 3)

    # Sai số bình phương trung bình căn

    rmse = round(np.sqrt(mean\_squared\_error(y\_test, y\_pred\_all)), 3)

    # Hệ số xác định R

    r2 = round(r2\_score(y\_test, y\_pred\_all), 3)

    # Sai số tuyệt đối trung bình

    mae = mean\_absolute\_error(y\_test, y\_pred\_all)

    # Lưu kết quả vào DataFrame

    metrics = pd.DataFrame({

        'Model Name': model\_name,

        'Accuracy(%)': [score],

        'Mean Squared Error (MSE)': [mse],

        'Root Mean Squared Error (RMSE)': [rmse],

        'R-squared (R2-Score)': [r2],

        'Mean Absolute Error (MAE)': [mae]

    })

    # Hàm trả về một DataFrame pandas

    return metrics

# Tính toán hiệu suất của các model

r1 = evaluate\_pipeline(lr, y\_pred\_lr, 'Linear Regression')

r2 = evaluate\_pipeline(dt1,y\_pred\_dt, 'Decision Tree - Best Tuned')

r3 = evaluate\_pipeline(dt2,y\_pred, 'Decision Tree - Basic Tuned')

r4 = evaluate\_pipeline(rf, y\_pred\_rf, 'Random Forest')

r5 = evaluate\_pipeline(ls, y\_pred\_ls, 'Lasso Regression')

r6 = evaluate\_pipeline(XGB, y\_pred\_xgb, 'XGBoost')

r7 = evaluate\_pipeline(abr, y\_pred\_abr, 'AdaBoost')

r8 = evaluate\_pipeline(gb, y\_pred\_gb, 'Gradient Boost')

r9 = evaluate\_pipeline(cat, y\_pred\_cat, 'CatBoost Regressor')

# Thêm các biến kết quả vào df all\_results

all\_results = pd.concat([r1, r2, r3, r4, r5, r6, r7, r8, r9], ignore\_index=True)

# Thêm results ( model Deep Learning) vào all\_results

all\_results = pd.concat([all\_results, result], ignore\_index=True)

all\_results

# Residual error là sự khác biệt giữa giá trị thực tế và giá trị dự đoán.

# Tính giá trị residual của các

residual\_lr = y\_test - y\_pred\_lr

residual\_dt = y\_test - y\_pred\_dt

residual = y\_test - y\_pred

residual\_rf = y\_test - y\_pred\_rf

residual\_ls = y\_test - y\_pred\_ls

residual\_xgb = y\_test - y\_pred\_xgb

residual\_abr = y\_test - y\_pred\_abr

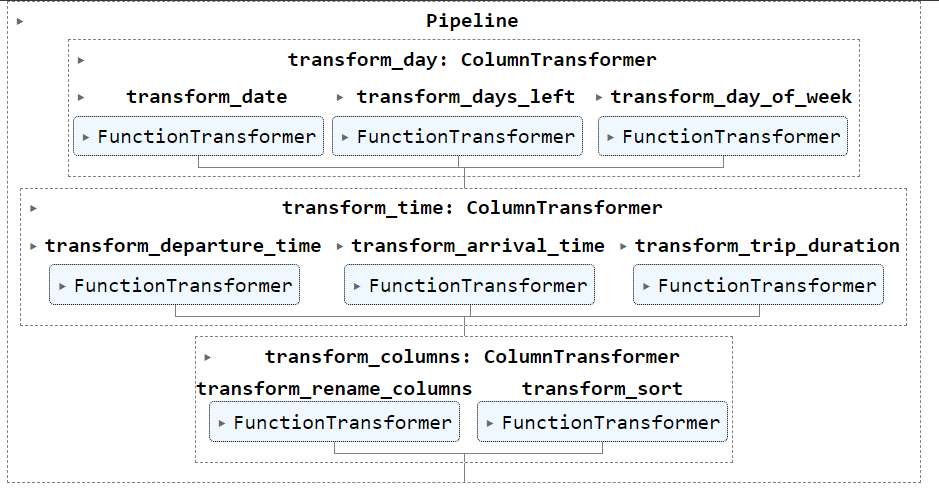
residual\_gb = y\_test - y\_pred\_gb

residual\_cat = y\_test - y\_pred\_cat

# **CHƯƠNG 5: KẾT QUẢ VÀ ĐÁNH GIÁ**

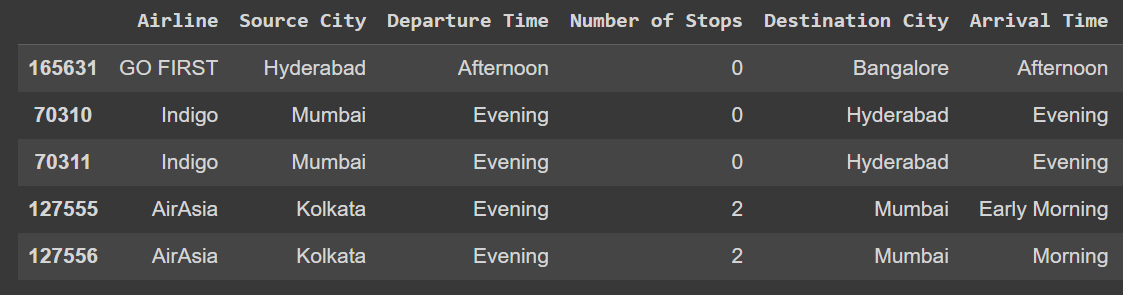
## 5.1 PIPELINE TRỰC QUAN HÓA SỰ CHUYỂN ĐỔI

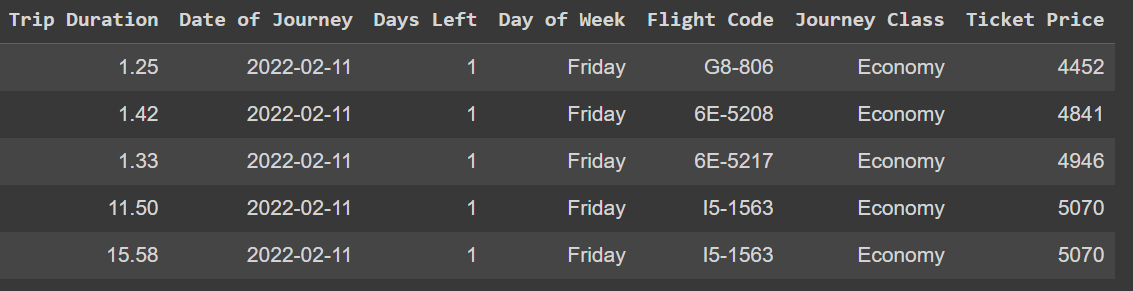
* Kết quả



## 5.2 TIỀN XỬ LÝ DỮ LIỆU

* Kết quả

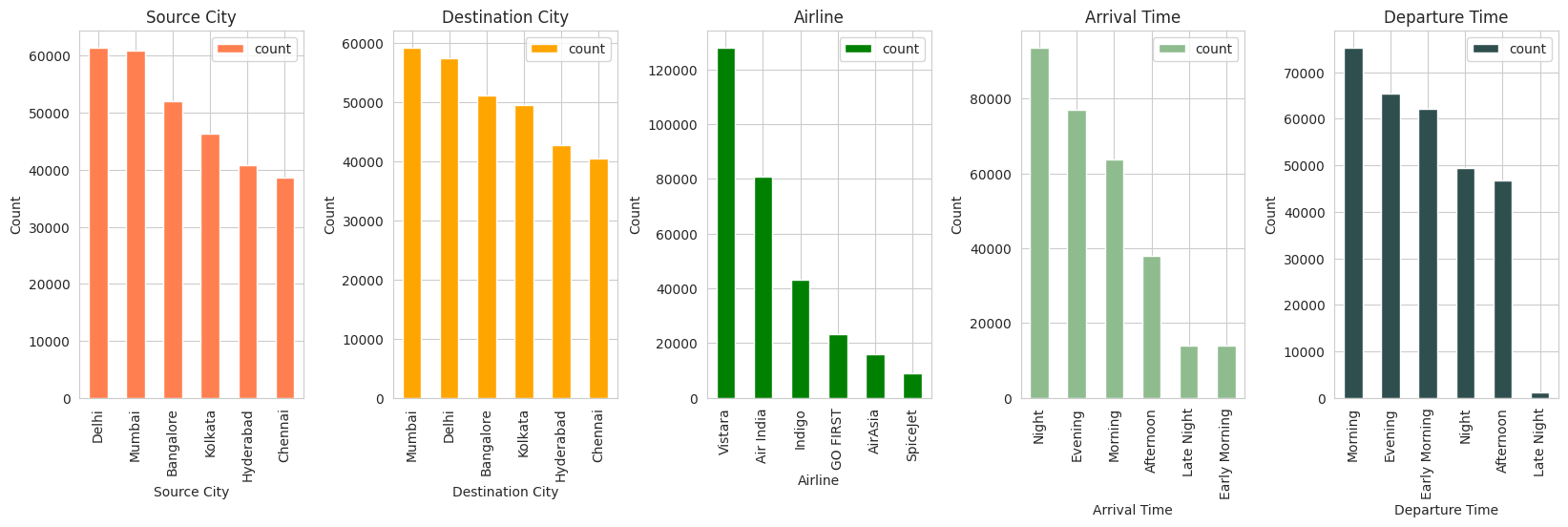




## 5.3 ĐỒ THỊ

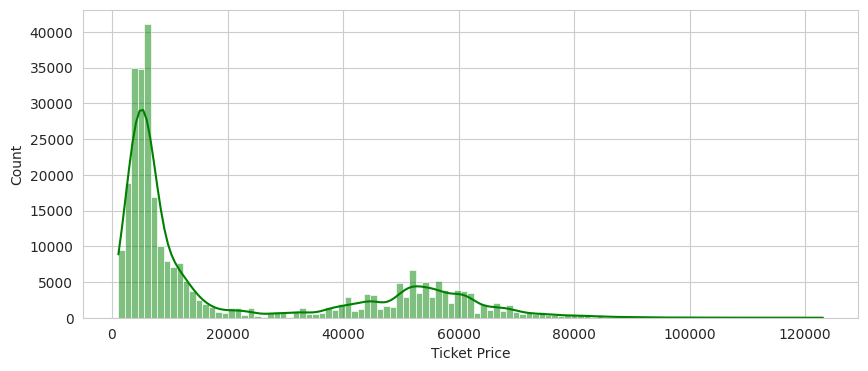
### 5.3.1 ĐỒ THỊ CÁC GIÁ TRỊ PHÂN LOẠI

* Kết quả



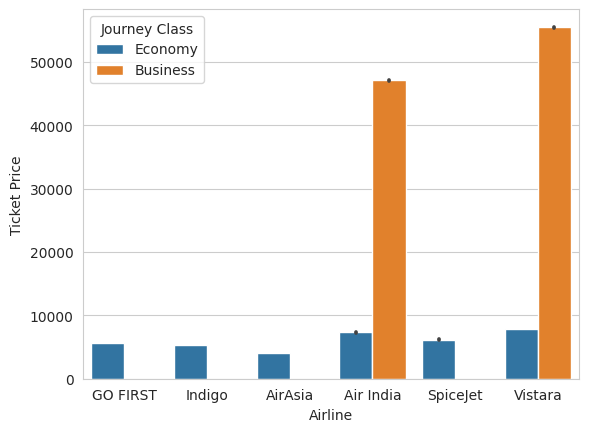
### 5.3.2 ĐỒ THỊ GIÁ VÉ VỚI SỐ LƯỢNG TƯƠNG ỨNG

* Kết quả



### 5.3.3 ĐỒ THỊ THỂ HIỆN SỰ THAY ĐỔI CỦA GIÁ VÉ THEO CÁC HÃNG HÀNG KHÔNG

* Kết quả



Từ Đồ thị ta thấy rằng, ở Hạng Phổ thông, Vistara là đắt nhất và Air Asia là rẻ nhất

Trong khi ở Hạng Thương gia, Vistara là đắt nhất và Air India là rẻ nhất

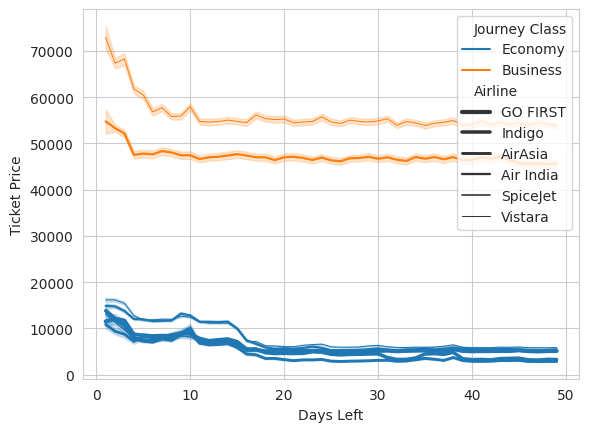
### 5.3.4 ĐỒ THỊ THỂ HIỆN SỰ ẢNH HƯỞNG CỦA THỜI GIAN MUA VÉ VỚI GIÁ VÉ

* Kết quả

Dựa trên phân tích dữ liệu, rõ ràng là giá vé máy bay tăng đáng kể khi được mua chỉ 1-2 ngày trước ngày khởi hành theo lịch trình, trong đó hạng Thương gia có thể tăng giá cao hơn so với vé hạng Phổ thông.

Hơn nữa, dữ liệu cho thấy giá vé tương đối ổn định trong một thời gian dài trước khi tăng đột ngột khoảng 30-35% đối với vé hạng Thương gia của Vistara Airlines và khoảng 15% đối với Air\_India.

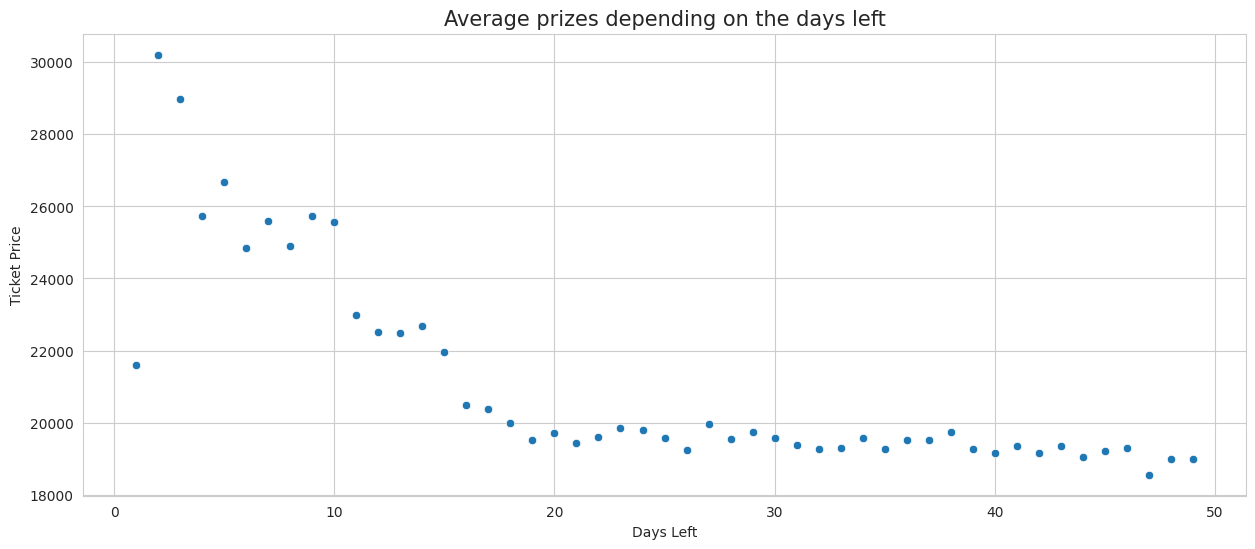
Để đảm bảo có được vé tiết kiệm chi phí nhất, nên mua vé hạng Phổ thông ít nhất ba tuần trước ngày khởi hành, trong khi đối với vé hạng Thương gia, nên mua vé ít nhất mười ngày trước ngày khởi hành theo lịch trình.



Dựa trên phân tích dữ liệu, rõ ràng là giá vé máy bay tăng đáng kể khi được mua chỉ 1-2 ngày trước ngày khởi hành theo lịch trình, trong đó hạng Thương gia có thể tăng giá cao hơn so với vé hạng Phổ thông.

Hơn nữa, dữ liệu cho thấy giá vé tương đối ổn định trong một thời gian dài trước khi tăng đột ngột khoảng 30-35% đối với vé hạng Thương gia của Vistara Airlines và khoảng 15% đối với Air\_India.

Để đảm bảo có được vé tiết kiệm chi phí nhất, nên mua vé hạng Phổ thông ít nhất ba tuần trước ngày khởi hành, trong khi đối với vé hạng Thương gia, nên mua vé ít nhất mười ngày trước ngày khởi hành theo lịch trình.



Có thể thấy hai đường cong khác nhau trên biểu đồ này, đường cong thứ nhất ổn định trong khoảng từ 50 đến 20 ngày trước chuyến bay và một đường cong đơn điệu dương trong khoảng từ 20 đến 2 ngày trước đó.

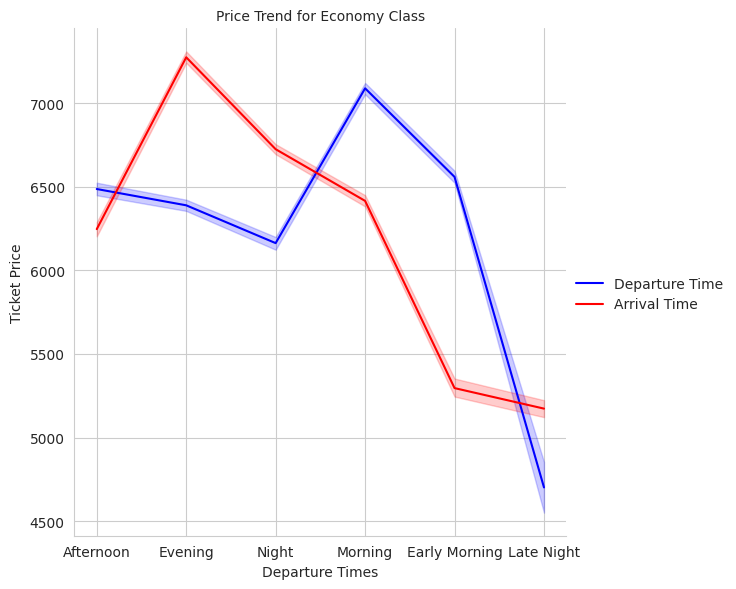


Có thể nhìn thấy rõ ràng trong cách giá phát triển tùy thuộc vào số ngày còn lại.

Biểu đồ cho thấy mức giá tăng chậm và sau đó bắt đầu tăng mạnh 20 ngày trước chuyến bay, nhưng giảm chỉ một ngày trước chuyến bay với giá rẻ hơn tới ba lần. 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.

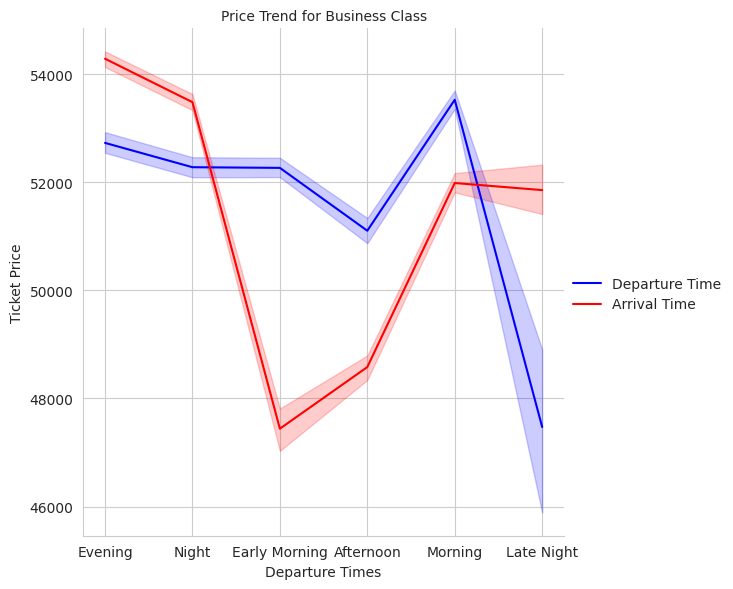
### 5.3.5 ĐỒ THỊ THỂ HIỆN SỰ THAY ĐỔI CỦA GIÁ VÉ VỚI THỜI GIAN KHỞI HÀNH VÀ THỜI GIAN ĐẾN

* Kết quả



Đối với hạng Phổ thông, Giá vé có xu hướng cao hơn khi Thời gian khởi hành là buổi sáng và giảm dần khi ngày trôi qua, chạm mức thấp nhất vào buổi tối trước khi tăng trở lại vào ban đêm.

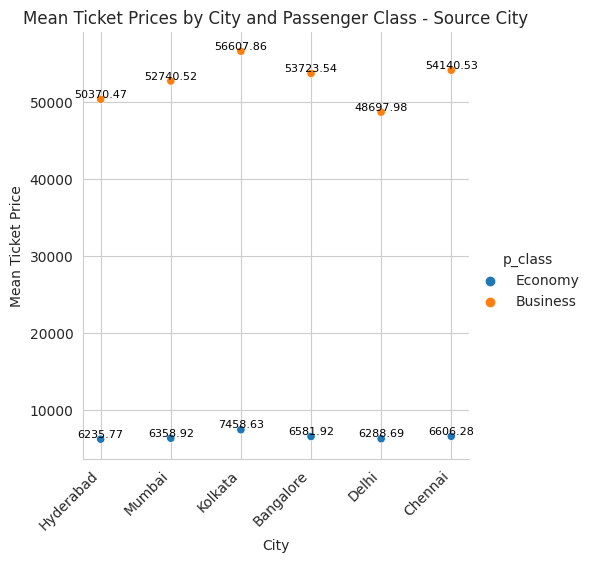
Tương tự như vậy, giá có xu hướng cao hơn khi thời gian đến là vào buổi chiều hoặc buổi tối, với giá thấp hơn vào buổi sáng hoặc buổi tối.



Xu hướng của hạng Thương gia hơi khác so với hạng Phổ thông, với giá vé có xu hướng cao nhất vào buổi chiều và buổi tối, và thấp hơn vào buổi sáng hoặc buổi tối.

### 5.3.6 ĐỒ THỊ THỂ HIỆN SỰ THAY ĐỔI CỦA GIÁ VÉ TÙY THUỘC VÀO NƠI CẤT CÁNH VÀ NƠI HẠ CÁNH

* Kết quả





Có thể thấy giá vé trung bình của Business cao hơn rất nhiều so với Economy ( gấp xấp xỉ 9 lần )

Giá trị trung bình của giá vé theo thành phố, cả 2 class thành phố Kolkata đều cao nhất, mức độ cao thấp của các thành phố trong 2 class cũng tương tự. Tuy nhiên thành phố thấp nhất của Business là Delhi, của Economy là Hyberabad



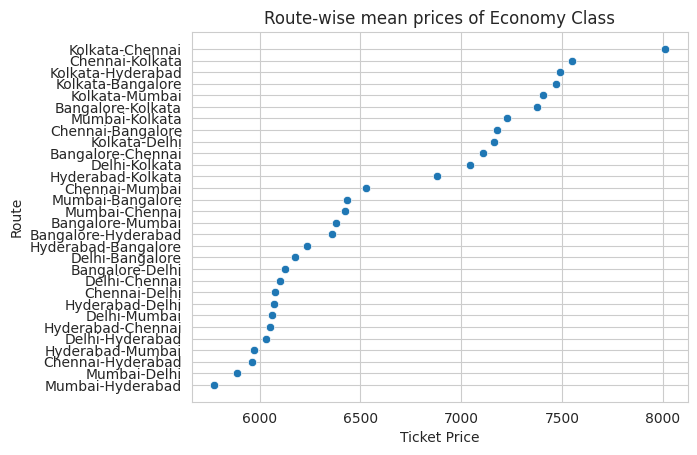


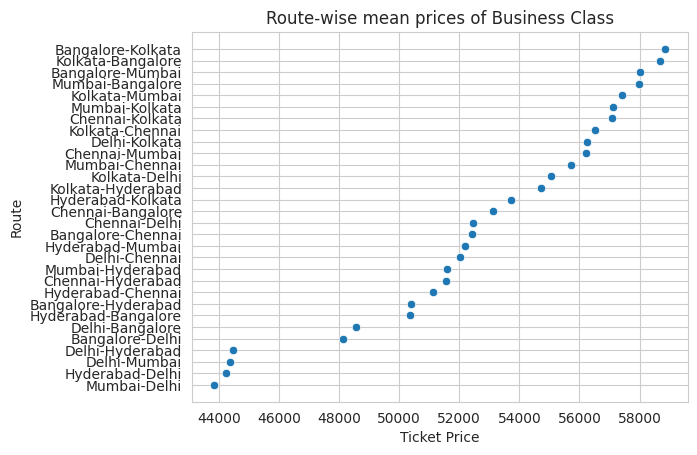
Có thể thấy giá vé trung bình của Business cao hơn rất nhiều so với Economy ( gấp xấp xỉ 9 lần )

Giá trị trung bình của giá vé theo thành phố, cả 2 class thành phố Kolkata đều cao nhất, thành phố Delhi đều thấp nhất, mức độ cao thấp của các thành phố trong 2 class cũng tương tự.

### 5.3.7 ĐỒ THỊ THỂ HIỆN GIÁ VÉ TRUNG BÌNH THEO LỘ TRÌNH CỦA TỪNG CLASS

* Kết quả





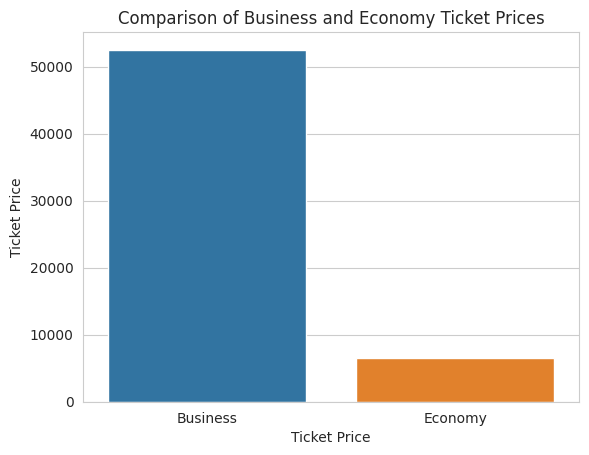
Dữ liệu cho thấy các chuyến bay khởi hành từ Delhi có vé rẻ nhất, tiếp theo là Hyderabad, Mumbai, Bangalore, Chennai và Kolkata, nơi vé cho các chuyến bay khởi hành đắt nhất. Tương tự, vé cho các chuyến bay đến Delhi là rẻ nhất, tiếp theo là Hyderabad, Mumbai, Bangalore, Chennai và Kolkata, nơi vé cho các chuyến bay đến đắt nhất.

Về giá vé trung bình cho các tuyến đường cụ thể, Chennai-Kolkata và Kolkata-Chennai có vé phổ thông đắt nhất trung bình, trong khi vé hạng thương gia cho Bangalore-Kolkata và Kolkata-Bangalore là đắt nhất. Điều này thật thú vị vì nó không có ý nghĩa về mặt địa lý và có thể là do lý do hậu cần.

Đối với hạng phổ thông, Mumbai-Hyderabad là tuyến rẻ nhất, tiếp theo là Mumbai-Delhi và Chennai-Hyderabad. Đối với hạng thương gia, Mumbai-Delhi là tuyến rẻ nhất, tiếp theo là Hyderabad-Delhi và Delhi-Mumbai.

### 5.3.8 ĐỒ THỊ THỂ HIỆN SỰ KHÁC NHAU GIỮA GIÁ VÉ HẠNG PHỔ THÔNG ( ECONOMY ) VÀ GIÁ VÉ HẠNG THƯƠNG GIÁ ( BUSINESS )

* Kết quả



Vé Hạng Thương gia thường đắt hơn vé Hạng Phổ thông. Điều này là do các tiện nghi và đặc quyền bổ sung đi kèm với Hạng Thương gia, chẳng hạn như nhiều chỗ để chân hơn, thức ăn ngon hơn và ưu tiên lên máy bay.

Tuy nhiên, vé Hạng Phổ thông đắt nhất từ Chennai đến Kolkata có giá khổng lồ 42349, cao hơn một số vé Hạng Thương gia cho các tuyến khác.

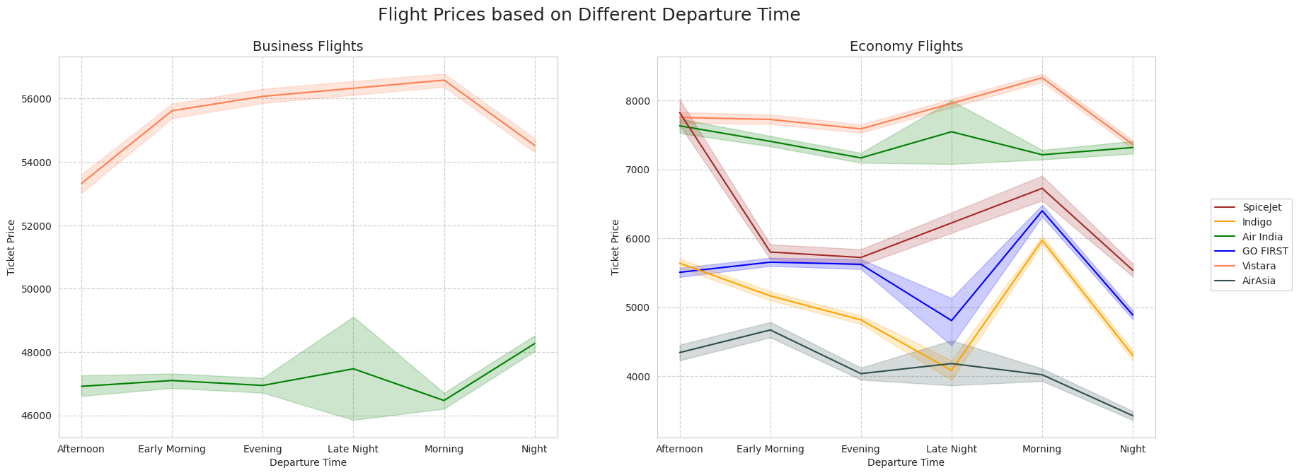
Mặt khác, vé Hạng Thương gia rẻ nhất cho chặng bay từ Bangalore đến Hyderabad chỉ có giá là 12000, thấp hơn đáng kể so với giá vé hạng Thương gia trung bình. Hiện tượng này có thể là do các yếu tố hậu cần và thị trường, chẳng hạn như sự cạnh tranh giữa các hãng hàng không và nhu cầu đối với các tuyến đường nhất định.

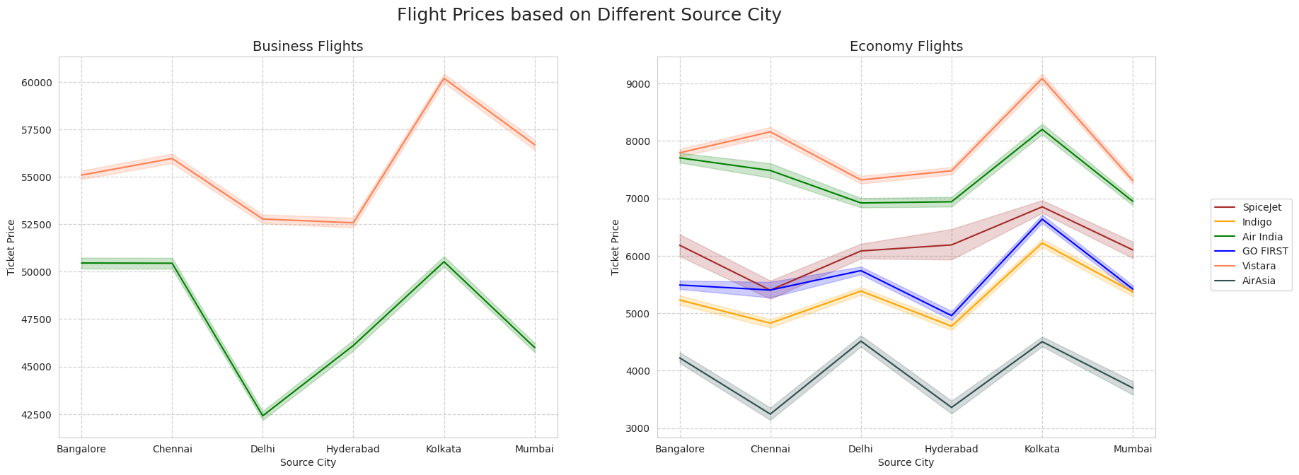
Ngoài ra, Air India dường như cung cấp cả vé hạng phổ thông đắt nhất và vé hạng thương gia rẻ nhất. Không có gì lạ khi các hãng hàng không có một số thay đổi về giá của họ và có thể chiến lược giá của Air India bao gồm việc cung cấp cả vé hạng cao vé phổ thông có giá và vé thương gia có giá thấp. Có thể có một số lý do cho chiến lược giá này. Ví dụ: Air India có thể đang cố gắng thu hút nhiều khách hàng hơn đến ghế hạng thương gia của mình bằng cách cung cấp cho họ mức giá thấp hơn. Đồng thời, hãng hàng không có thể đang tìm cách tối đa hóa doanh thu bằng cách tính phí bảo hiểm cho các vé hạng phổ thông đắt nhất của mình. Cũng có thể những thay đổi về giá này là kết quả của các yếu tố cung và cầu, chẳng hạn như tình trạng còn chỗ trên các chuyến bay cụ thể hoặc mức độ phổ biến của một số tuyến bay nhất định. Nhìn chung, thật khó để nói chắc chắn tại sao Air India lại cung cấp cả vé hạng phổ thông giá cao và vé hạng thương gia giá rẻ mà không có thêm thông tin về chiến lược giá và các yếu tố thị trường của họ.

Nhìn chung, trong khi chênh lệch giá giữa vé Hạng Thương gia và Hạng Phổ thông có thể là xu hướng chung, vẫn có thể có những ngoại lệ đáng chú ý đáng để khám phá và tìm hiểu trong bối cảnh của ngành hàng không.

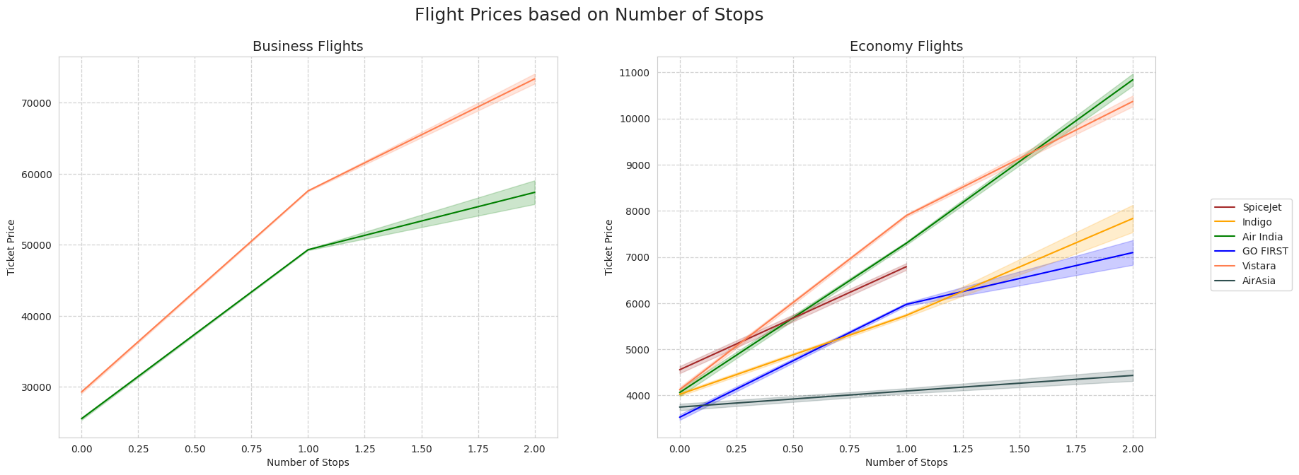
### 5.3.9 ĐỒ THỊ THỂ HIỆN GIÁ VÉ VỚI CÁC THÔNG SỐ KHÁC NHAU THEO TỪNG CLASS

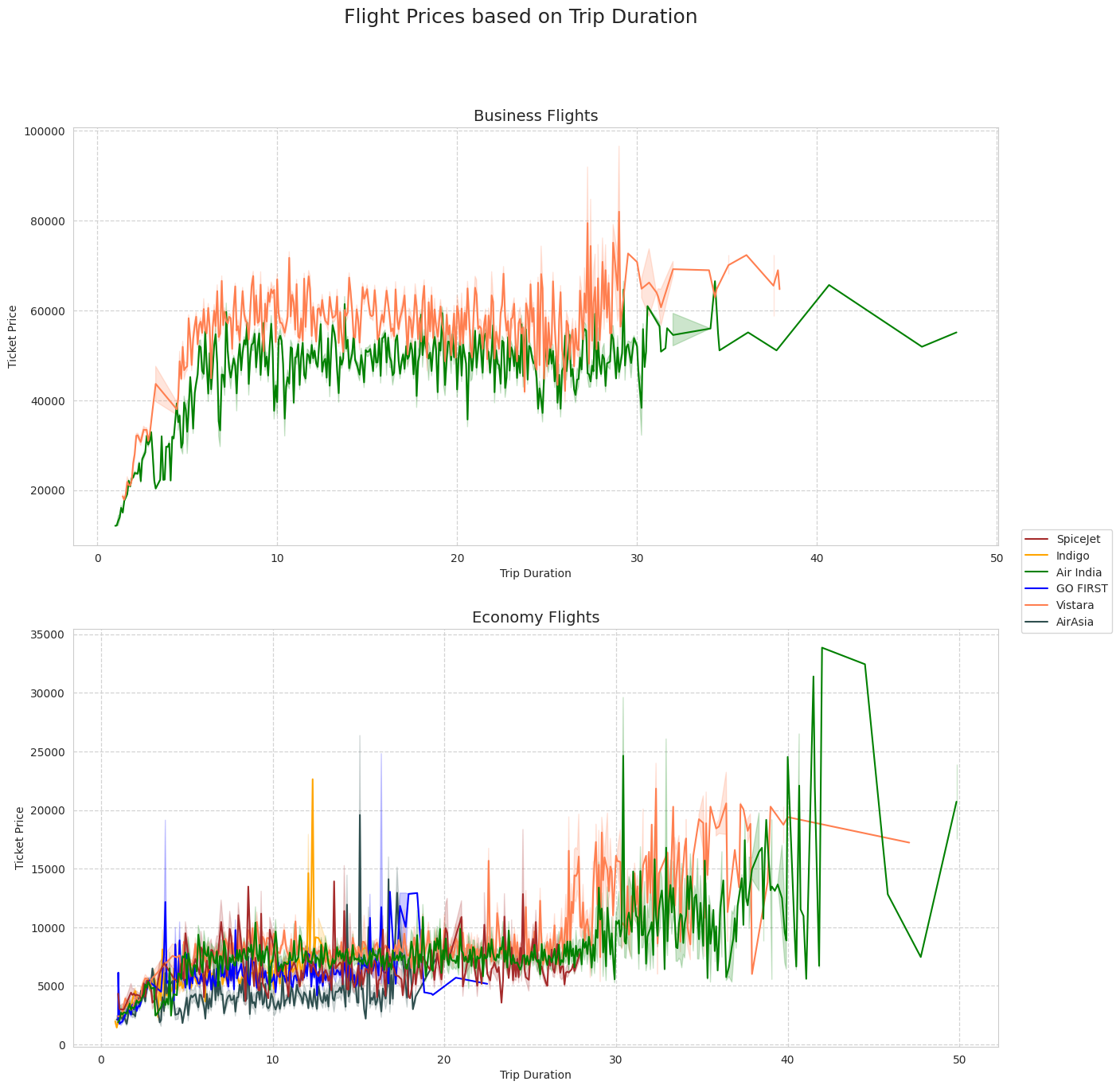
* Kết quả

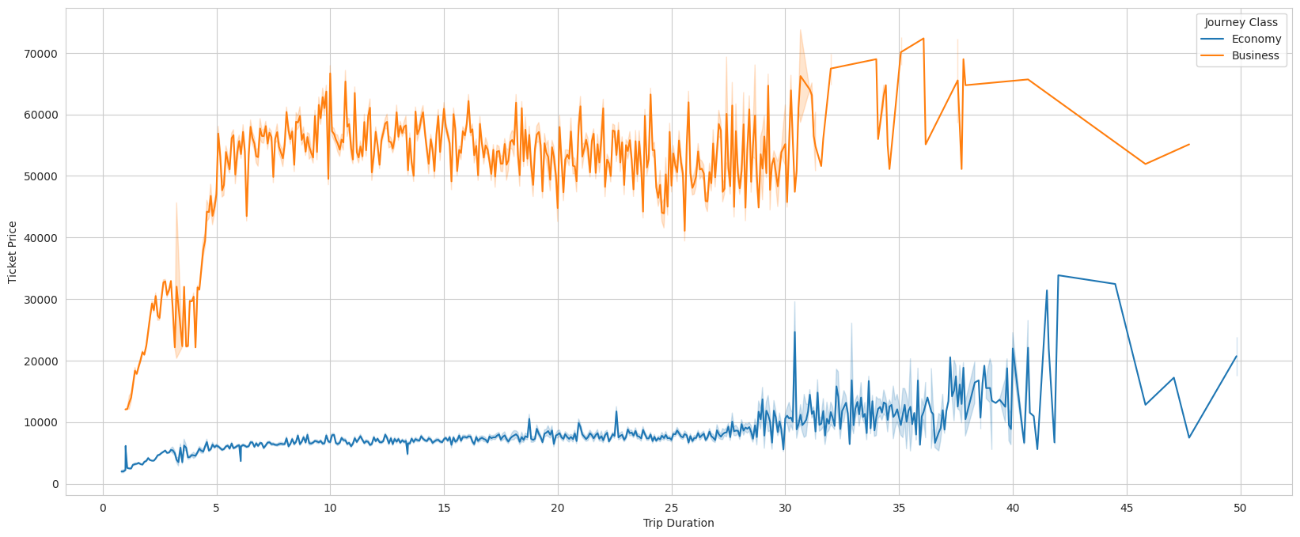


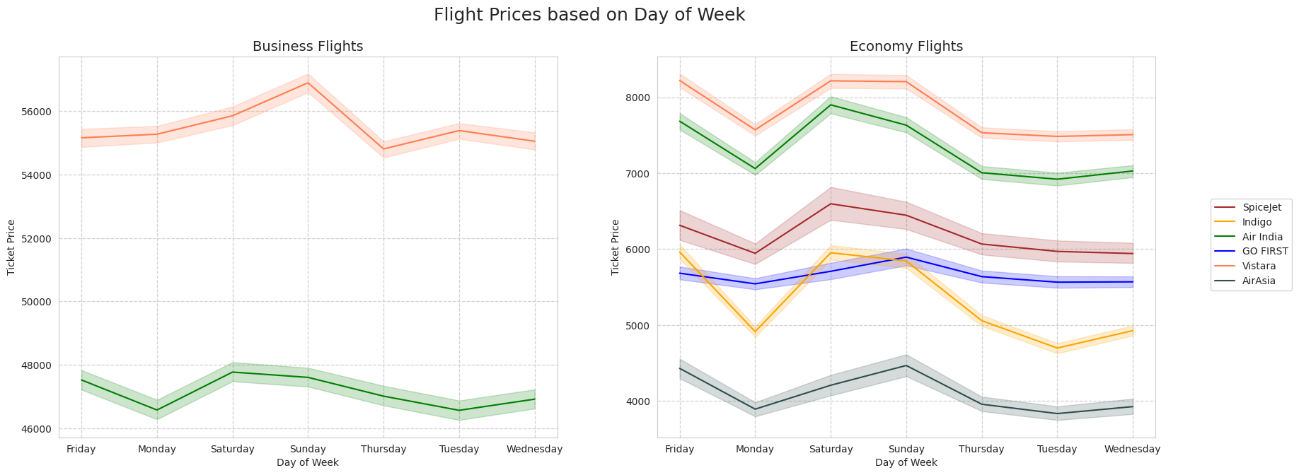


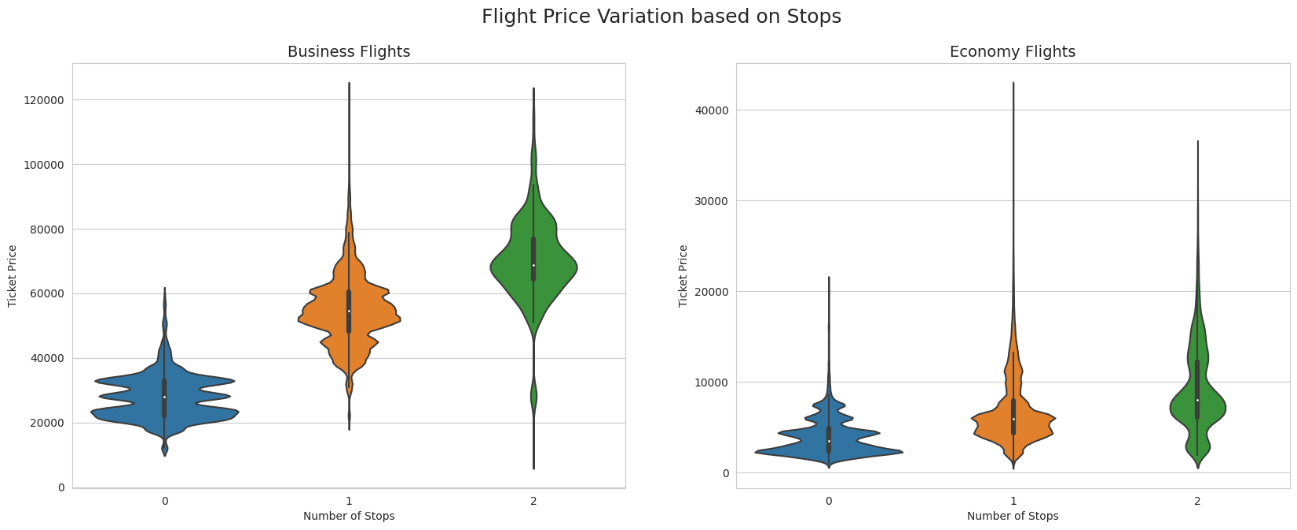


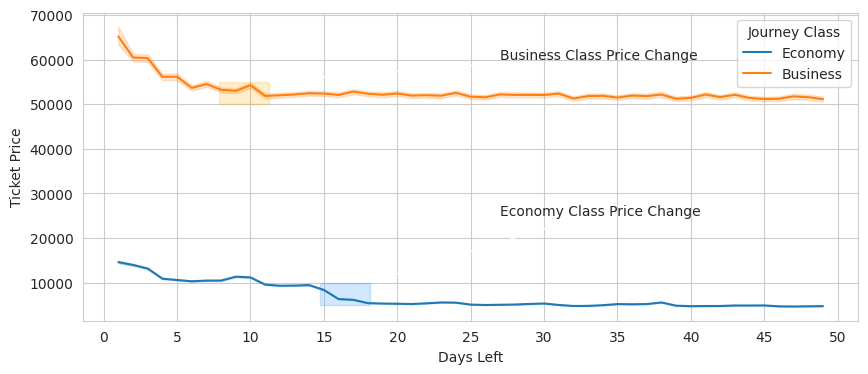






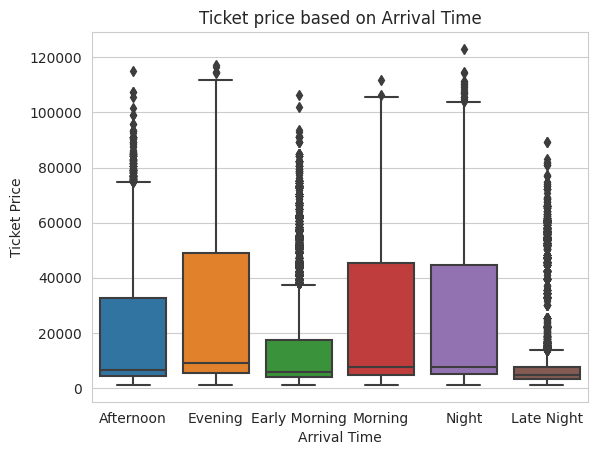






### 5.3.10 ĐỒ THỊ T-TEST THỂ HIỆN SỰ THAY ĐỔI CỦA GIÁ VÉ PHỤ THUỘC VÀO THỜI GIAN KHỞI HÀNH VÀ THỜI GIAN ĐẾN

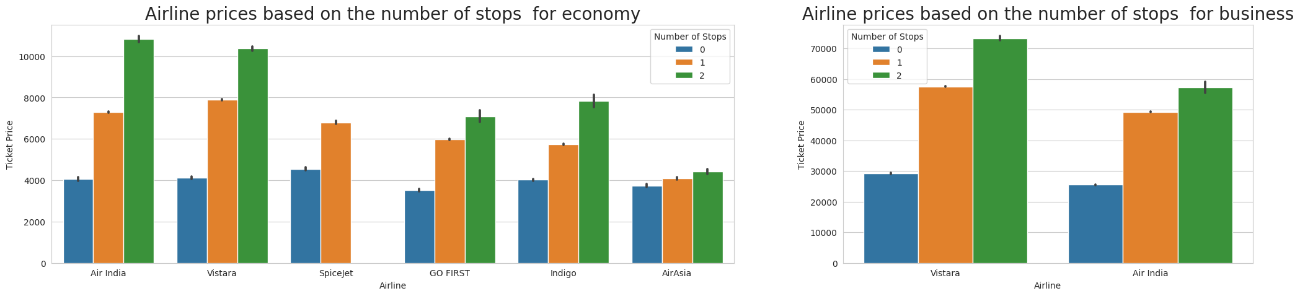
* Kết quả



Theo dự kiến, khởi hành vào ban đêm hoặc đến vào ban đêm vẫn là cách rẻ nhất để đi du lịch. Nhưng cũng có thể thấy là đến sáng sớm cũng rẻ và bay chiều rẻ hơn một chút là bay chiều, sáng và tối.

### 5.3.11 ĐỒ THỊ THỂ HIỆN SỰ ẢNH HƯỞNG CỦA SỐ ĐIỂM DỪNG CỦA TỪNG HÃNG HÀNG KHÔNG ĐẾN GIÁ VÉ

* Kết quả



Rõ ràng là càng có 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.

### 5.3.12 ĐỒ THỊ THỂ HIỆN SỰ ẢNH HƯỞNG CỦA THỜI GIAN BAY ĐẾN GIÁ VÉ – MEAN/ MIN/ MAX PRICE

* Kết quả



Trung bình, các chuyến bay có thời gian dài hơn có xu hướng đắt hơn các chuyến bay có thời gian vừa phải hoặc ngắn ngày, bất kể đó là hạng phổ thông hay hạng thương gia. Tuy nhiên, đối với các hành khách hạng phổ thông, các chuyến bay chặng ngắn có giá vé đắt hơn một cách đáng ngạc nhiên so với các chuyến bay chặng dài. Mặt khác, đối với hạng thương gia, các chuyến bay có khoảng cách vừa phải thường có giá vé đắt hơn. Nhìn chung, những kết quả này cho thấy rằng thời lượng chuyến bay thực sự có tác động đến giá vé và điều quan trọng là phải xem xét yếu tố này khi lên kế hoạch cho một chuyến đi.

### 5.3.13 ĐỒ THỊ THỂ HIỆN SỰ ẢNH HƯỞNG CỦA SỐ ĐIỂM DỪNG ĐẾN GIÁ VÉ – MEAN/ MIN/ MAX PRICE

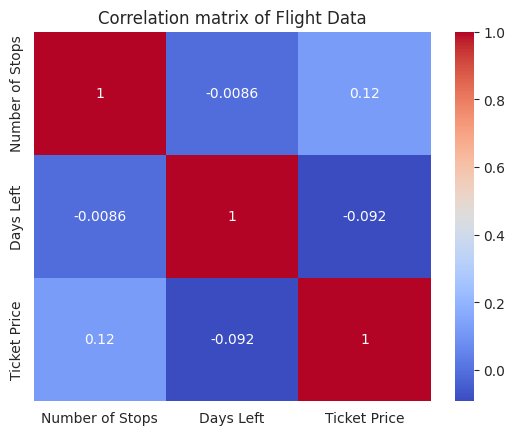
* Kết quả



Khi phân tích tác động của thời gian bay đến giá vé, cho thấy các chuyến bay có hai điểm dừng trở lên có mức giá trung bình cao nhất, tiếp theo là các chuyến bay một điểm dừng và các chuyến bay thẳng. Tuy nhiên, các chuyến bay một điểm dừng có giá vé tối đa cao nhất, trong khi các chuyến bay hai điểm dừng trở lên và các chuyến bay thẳng có giá tối đa thấp hơn. Mặt khác, các chuyến bay thẳng có mức giá tối thiểu thấp nhất, trong khi các chuyến bay một điểm dừng và hai điểm dừng trở lên có mức giá tối thiểu cao hơn. Quan sát cả hạng phổ thông và hạng thương gia, cho thấy một mô hình thống nhất. Nhìn chung, thời lượng chuyến bay đóng một vai trò quan trọng trong việc xác định giá vé, với các chuyến bay dài hơn thường có giá cao hơn, đặc biệt là khi có thêm các điểm dừng.

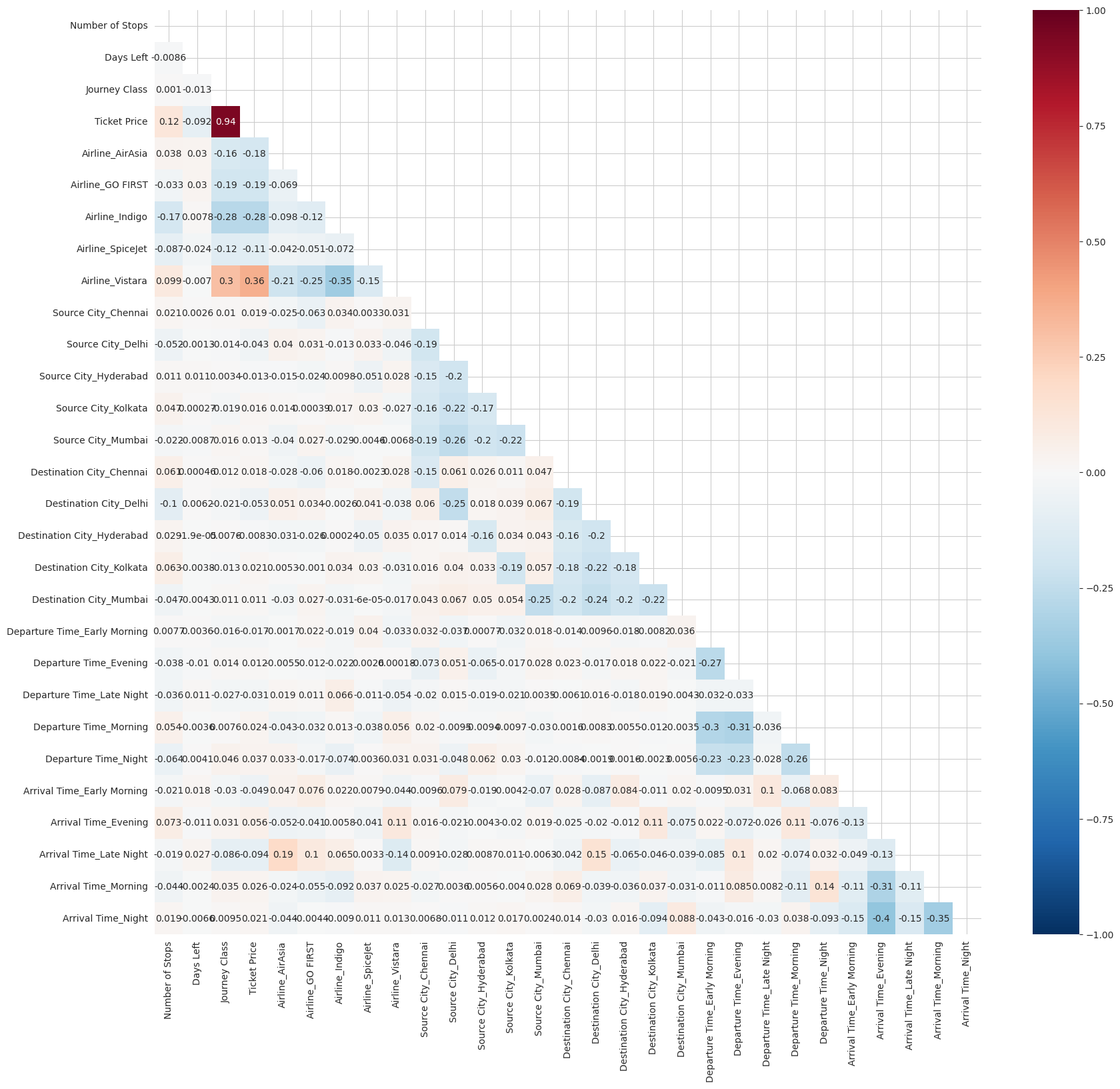
### 5.3.14 MATRIX TƯƠNG QUAN CỦA CÁC CỘT SỐ

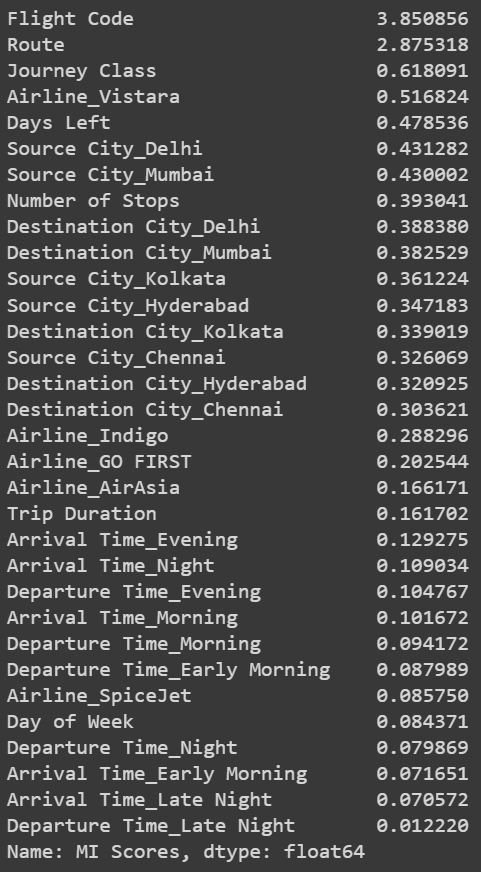
* Kết quả



### 5.3.15 ĐỒ THỊ TỔNG QUAN THỂ HIỆN SỰ ẢNH HƯỞNG CỦA CÁC BIẾN GIÁ TRỊ ĐẾN GIÁ VÉ

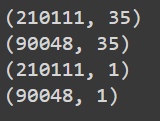
* Kết quả

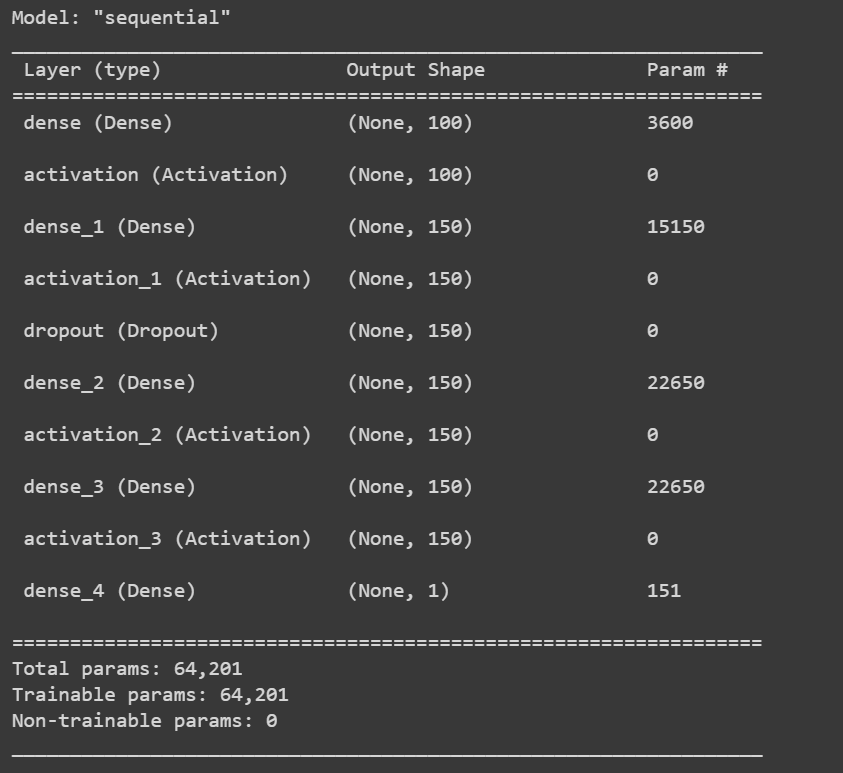
****

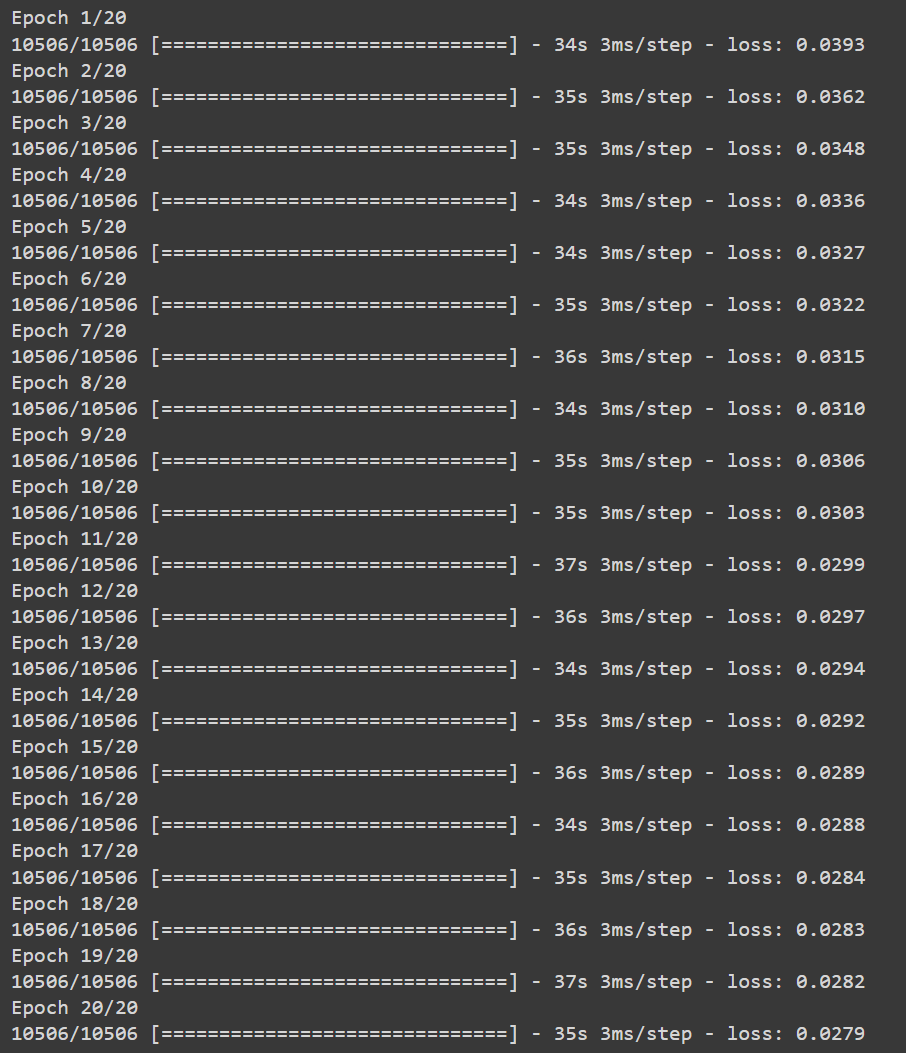


## 5.4 MÔ HÌNH ĐỊNH GIÁ VÉ MÁY BAY THEO YÊU CẦU NGƯỜI DÙNG

* Kết quả

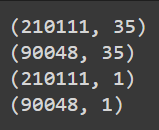


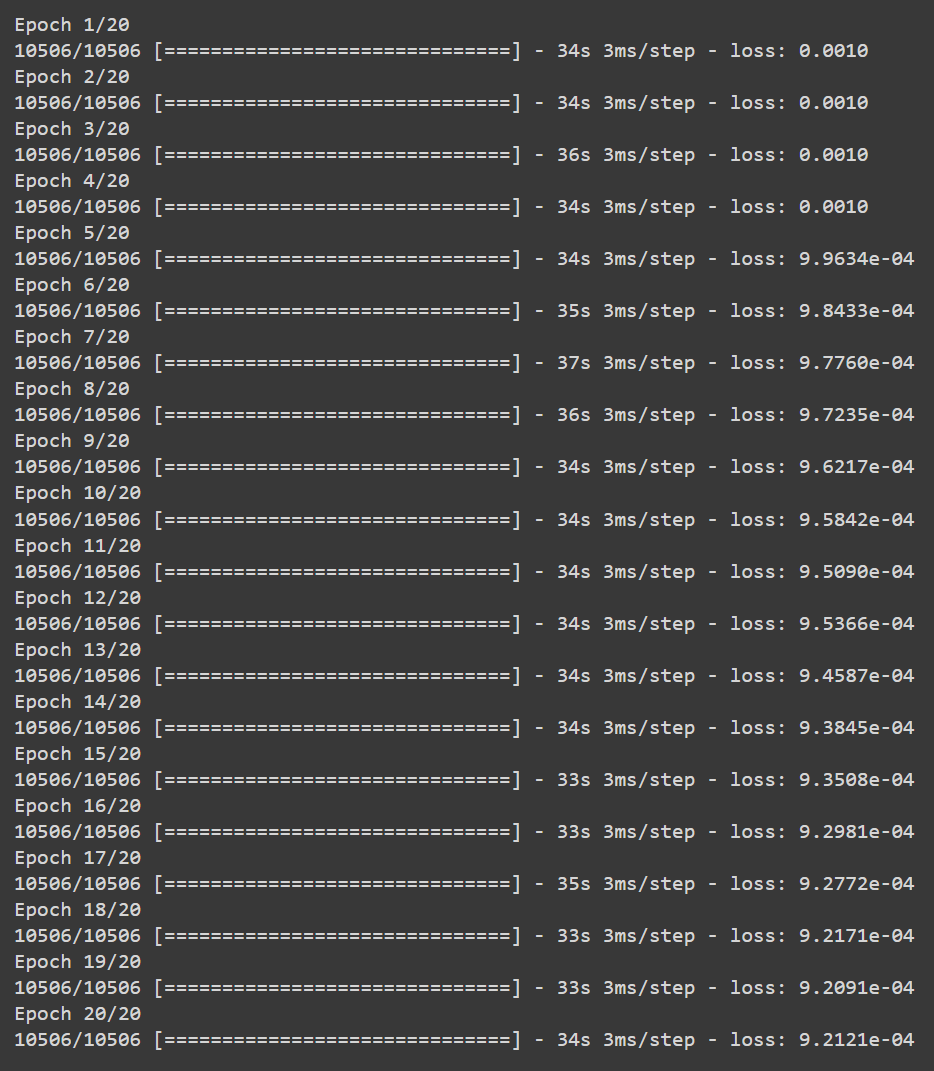
****

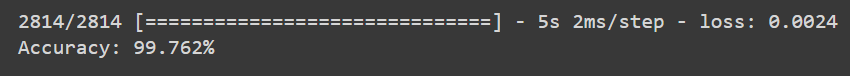


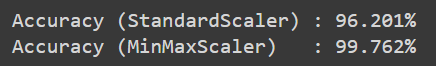


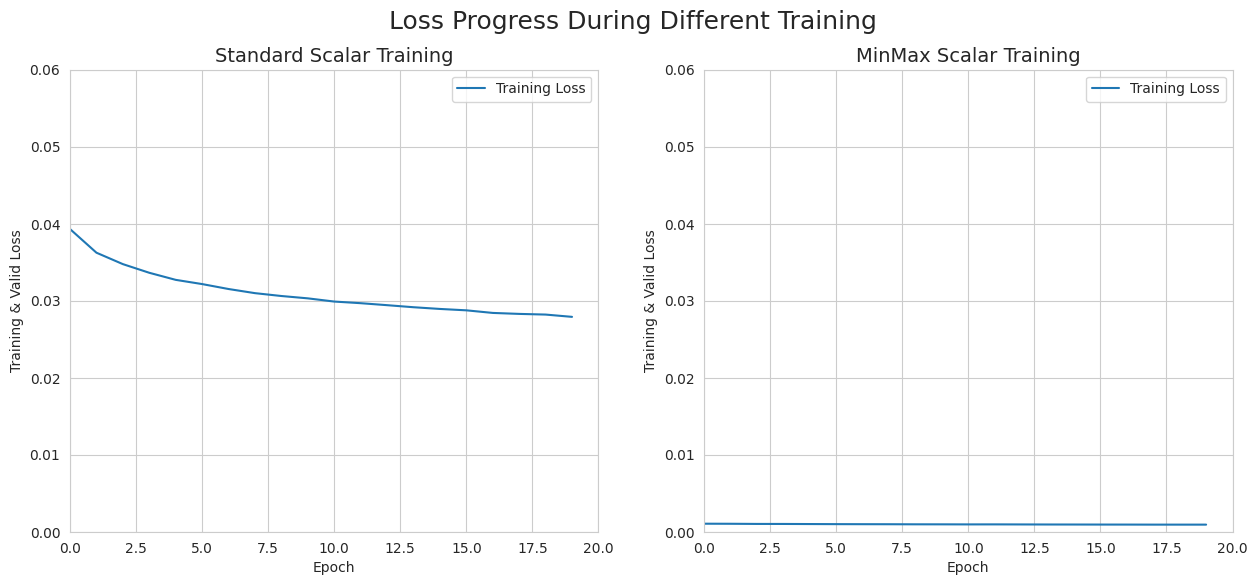


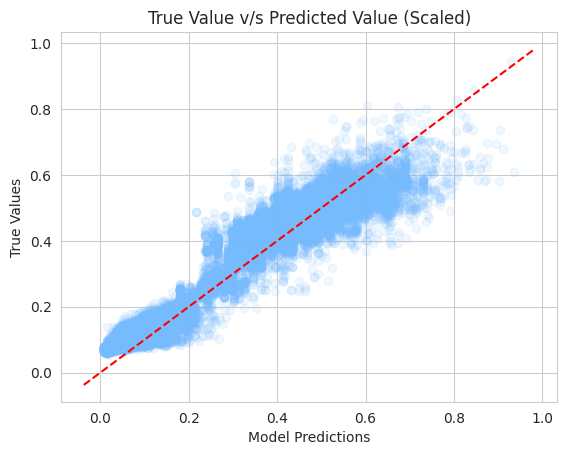


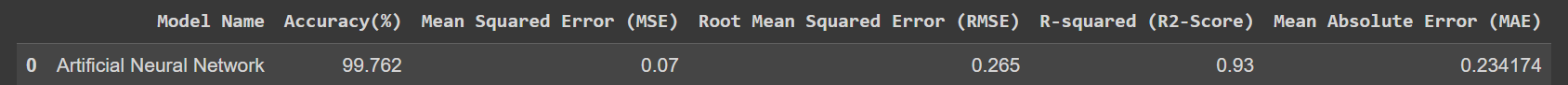






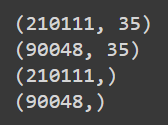


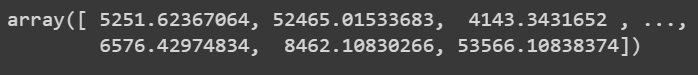


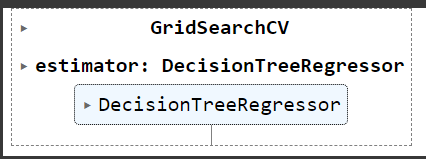


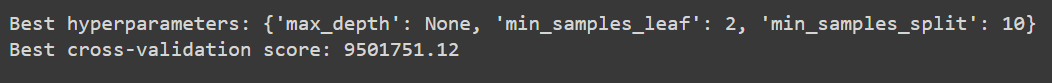
## 5.5 CÁC MÔ HÌNH DỰ ĐOÁN LIÊN QUAN

* Kết quả

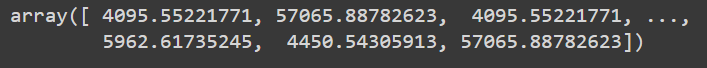


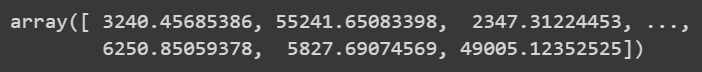


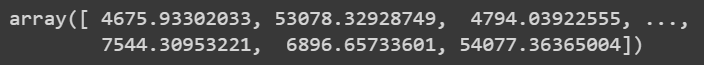


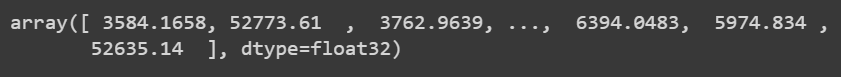


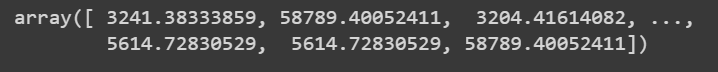


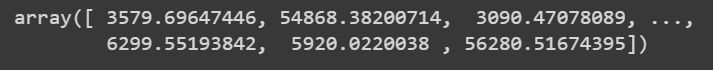


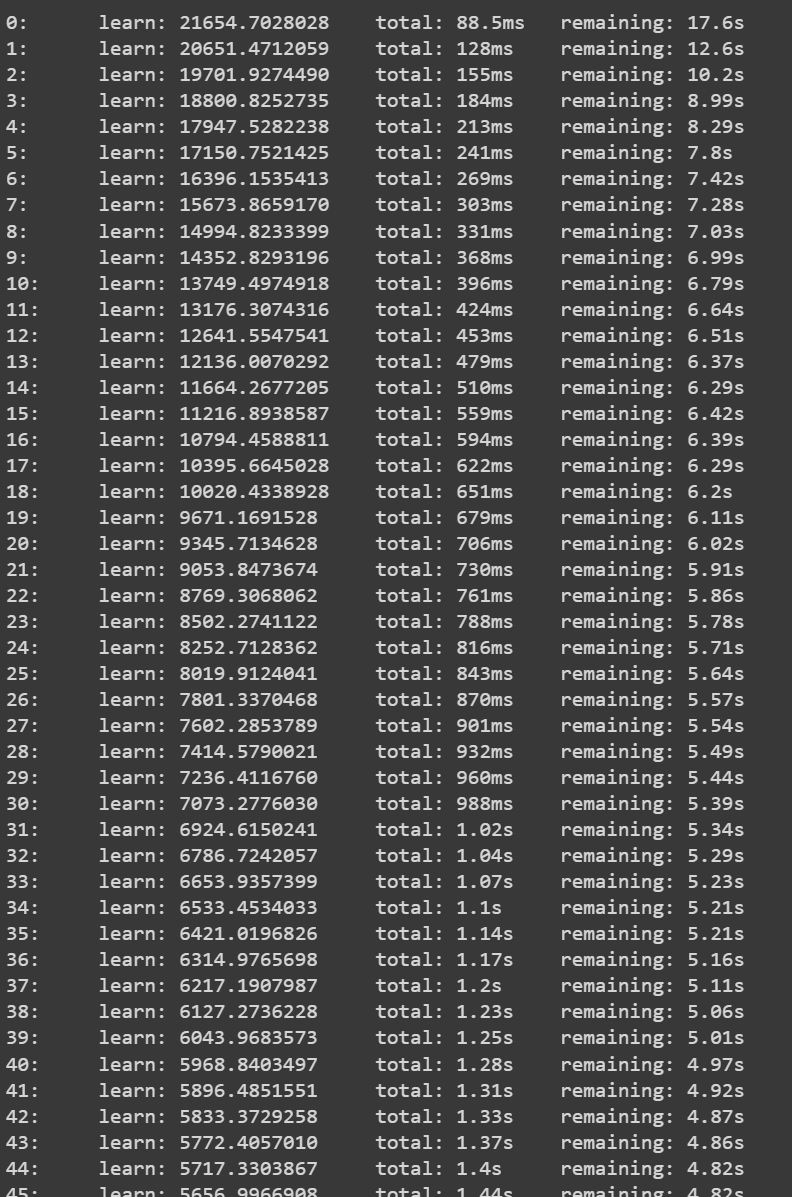


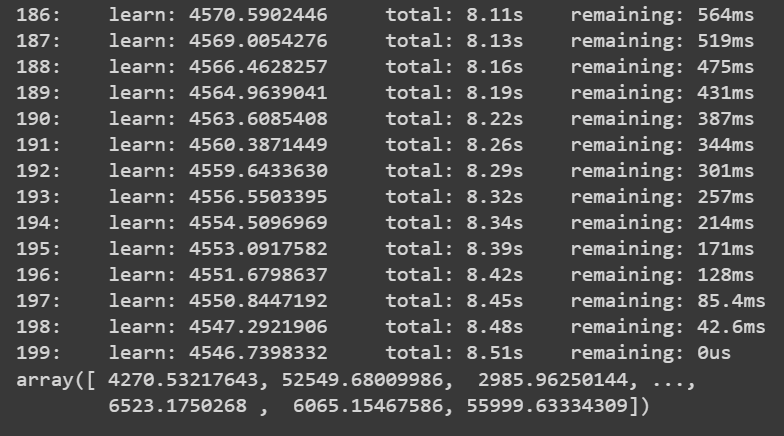


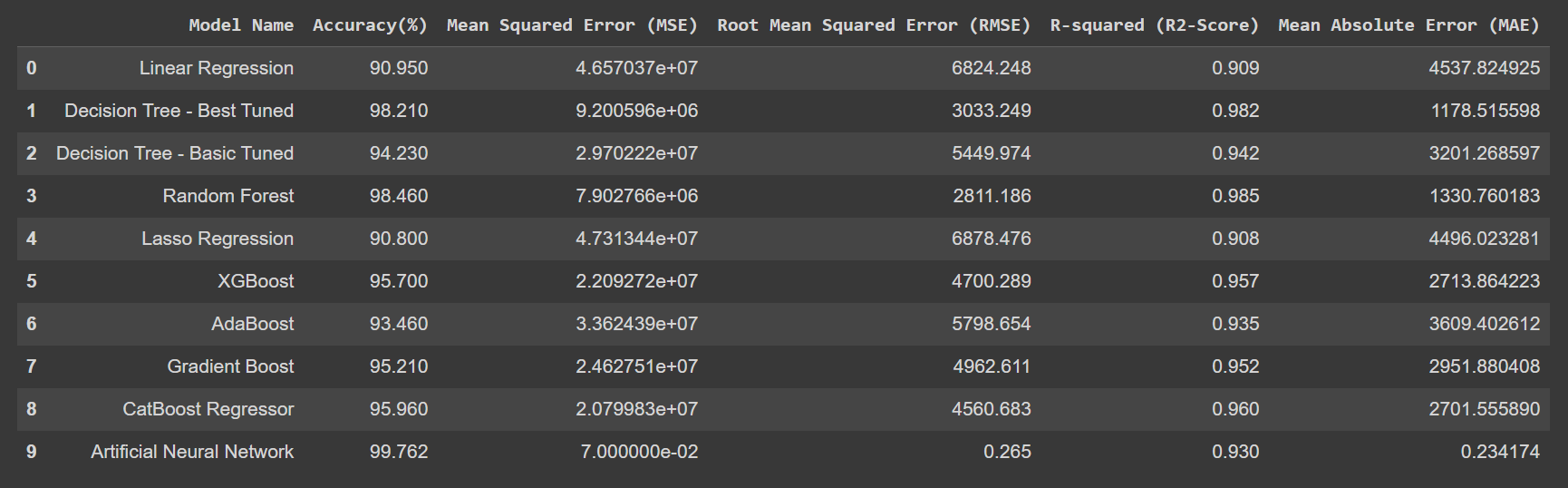


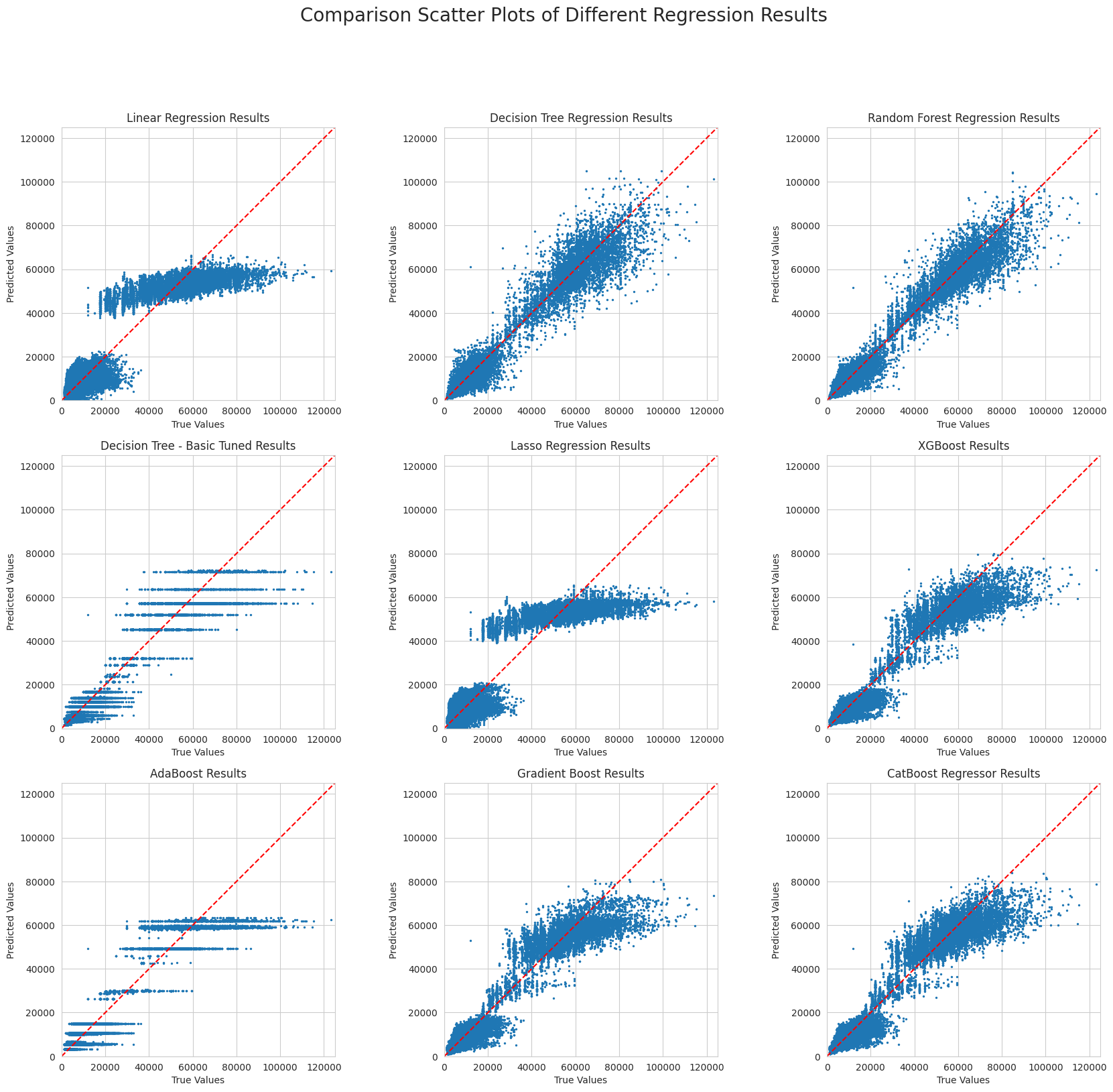


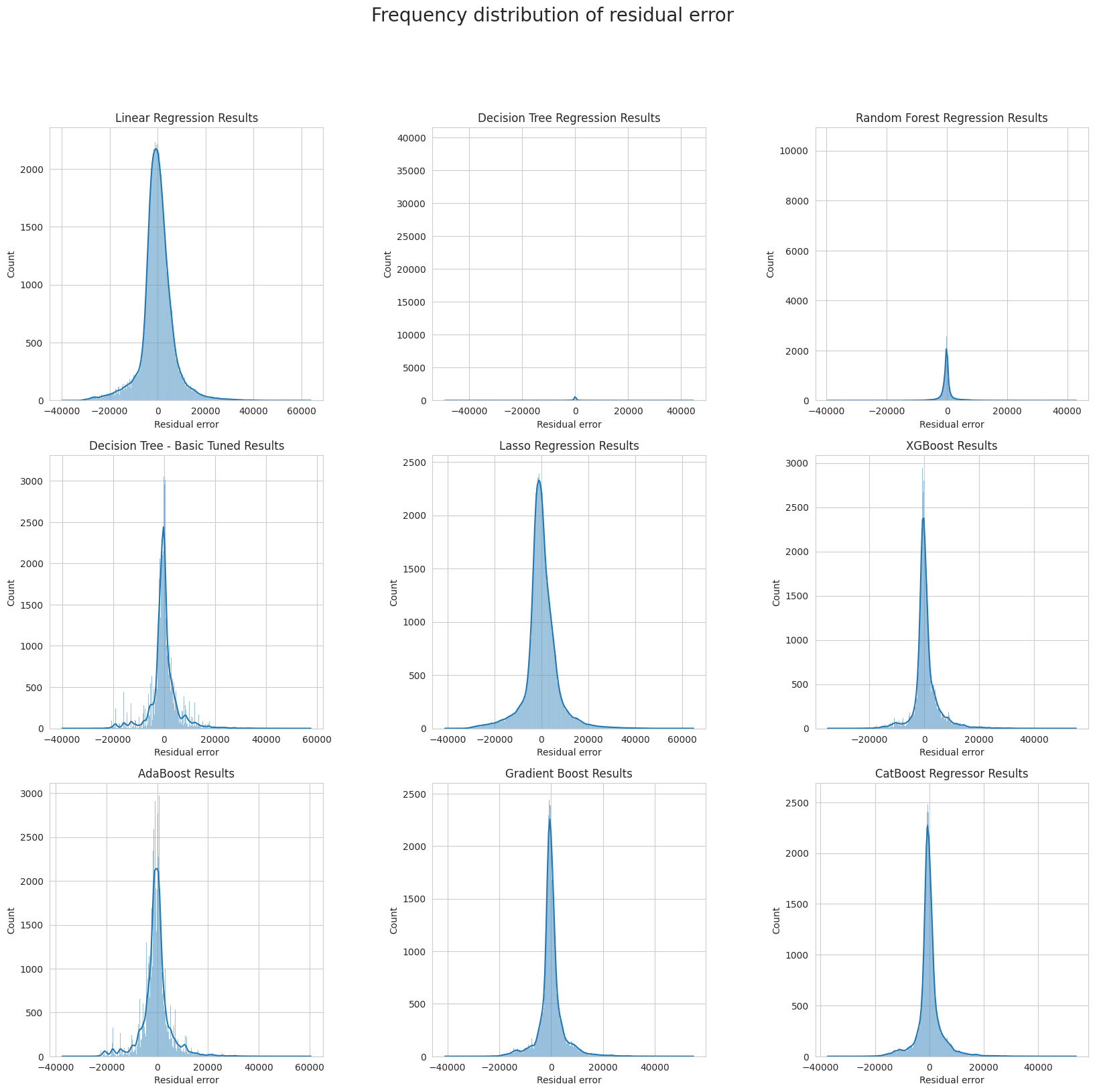












# **CHƯƠNG 6: KẾT LUẬN VÀ ĐỊNH HƯỚNG PHÁT TRIỂN**

## 6.1 KẾT LUẬN

Nếu muốn tiết kiệm tiền mua vé máy bay, có một số yếu tố cần xem xét trước khi mua hàng. Một trong những yếu tố quan trọng nhất là thời gian của các chuyến bay. Điều đáng chú ý là thời gian trong ngày và ngày trong tuần có thể có tác động đáng kể đến giá vé. Nhìn chung, các chuyến bay khởi hành và đến vào lúc đêm khuya thường rẻ hơn so với những chuyến bay khởi hành và đến vào giờ cao điểm. Đối với các chuyến bay có thời gian hành trình 2-3 giờ không điểm dừng, khởi hành vào đêm khuya và thời gian đến thường là một lựa chọn khả thi. Các chuyến bay này thường khởi hành sau 9 hoặc 10 giờ tối và đến điểm đến vào đầu giờ sáng. Mặc dù các chuyến bay này có thể không thuận tiện nhất về mặt lịch trình, nhưng chúng có thể là một cách tuyệt vời để tiết kiệm tiền mua vé.

Một yếu tố khác cần xem xét khi đặt chuyến bay là số điểm dừng. Trong một số trường hợp, chuyến bay có nhiều điểm dừng hơn có thể rẻ hơn chuyến bay có ít điểm dừng hơn, tùy thuộc vào đường bay và hãng hàng không. Nên thực hiện một số nghiên cứu để tìm ra hãng hàng không nào cung cấp các giao dịch tốt nhất cho tuyến đường đã chọn và họ thường thực hiện bao nhiêu điểm dừng.

Khi nói đến việc tìm kiếm ưu đãi tốt nhất cho vé máy bay, điều quan trọng là phải linh hoạt và cởi mở với các lựa chọn khác nhau. Bằng cách xem xét các yếu tố như thời gian, số điểm dừng và các tùy chọn hãng hàng không, bạn có thể tăng cơ hội tìm được ưu đãi lớn trên chuyến bay tiếp theo.

## 6.2 ĐỊNH HƯỚNG PHÁT TRIỂN

Để phát triển đề tài "Định giá vé máy bay theo lịch trình và loại bay môn" trong tương lai, có thể xem xét các hướng đi sau đây:

1. Mở rộng phạm vi dữ liệu: Tiếp tục thu thập và tích hợp các nguồn dữ liệu mới để tăng cường dữ liệu huấn luyện. Bao gồm việc thu thập thông tin về yếu tố khách hàng, thông tin về thời tiết, sự kiện địa phương, và các yếu tố khác có thể ảnh hưởng đến giá vé máy bay.

2. Nâng cao độ chính xác của mô hình: Nghiên cứu và áp dụng các thuật toán và phương pháp mới trong lĩnh vực khoa học dữ liệu để cải thiện độ chính xác của mô hình. Có thể thử nghiệm các mô hình học sâu (deep learning) hoặc kỹ thuật học tăng cường (reinforcement learning) để đạt được kết quả tốt hơn.

3. Tối ưu hóa giá vé dựa trên thông tin đa chiều: Nghiên cứu cách sử dụng thông tin lịch trình và loại chuyến bay để tối ưu hóa giá vé trong thời gian thực. Điều này có thể bao gồm việc xây dựng một hệ thống định giá động (dynamic pricing) để điều chỉnh giá vé dựa trên các yếu tố biến đổi như ngày, thời gian, tỷ lệ đặt chỗ, và nhu cầu của thị trường.

4. Tích hợp các yếu tố xã hội và kỹ thuật số: Khai thác sử dụng dữ liệu từ các mạng xã hội và nguồn thông tin kỹ thuật số khác để cải thiện định giá vé máy bay. Ví dụ, phân tích tư duy người dùng và phản hồi xã hội có thể cung cấp thông tin giá trị về sự ưa chuộng và đánh giá của khách hàng đối với các chuyến bay cụ thể.

5. Phát triển ứng dụng thực tế: Xây dựng các ứng dụng hoặc công cụ trực tuyến cho phép khách hàng tìm kiếm, so sánh và đặt vé máy bay dựa trên thông tin lịch trình và loại chuyến bay. Đồng thời, tối ưu hóa quy trình đặt vé và tạo ra trải nghiệm mua vé trực tuyến thuận tiện và thông minh hơn.

6. Đánh giá tác động và phản hồi thị trường: Nghiên cứu tác động của mô hình định giá vé máy bay theo lịch trình và loại chuyến bay đối với thị trường hàng không và khách hàng. Điều này có thể bao gồm việc thu thập phản hồi từ khách hàng và các bên liên quan để cải thiện và điều chỉnh mô hình theo thời gian.

Tổng quan, việc phát triển đề tài này cần sự liên tục và cập nhật với xu hướng công nghệ và thị trường. Đồng thời, đòi hỏi sự hợp tác với các hãng hàng không, nhà quản lý và các chuyên gia trong lĩnh vực này để đạt được kết quả ứng dụng thực tế và có giá trị cao.

# **CHƯƠNG 7: NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN VÀ TIẾN HÀNH VẤN ĐÁP VỀ ĐỀ TÀI**

# **TÀI LIỆU THAM KHẢO**

* Tài liệu tham khảo về thuật toán và code

[1] Zhang, Y., Qin, X., Liu, X., & Lin, H. (2020). Airline ticket price forecasting using machine learning models. Journal of Air Transport Management, 87, 101865.

[2] Zeng, X., Liu, D., Ren, Y., & Yan, W. (2019). Airline ticket price prediction based on machine learning algorithms. IEEE Access, 7, 146488-146496.

[3] Bhushan, B., & Tomar, D. (2019). Airline ticket price prediction using machine learning techniques. In 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.

* Tài liệu tham khảo khác

[4] Chatterjee, P., & Roy, A. (2014). Prediction of airfare using regression analysis. International Journal of Computer Applications, 108(1), 1-4.

[5] Mathew, B., & Jacob, M. (2014). Forecasting domestic airfares using time series models and adaptive neural networks. Procedia Computer Science, 35, 201-210.