TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

BÁO CÁO ĐỒ ÁN 2

XÂY DỰNG VÀ KIỂM THỬ MÔ HÌNH HỌC MÁY CHO VIỆC DỰ ĐOÁN CHUYẾN BAY TRỄ GIỜ

Ngành: HỆ THỐNG THÔNG TIN QUẢN LÝ

Giảng viên:	ThS. NGUYỄN TUẤN DŨNG
Sinh viên thực hiện:	Lê Hữu Đức Long - 20185464
Lớp:	MI2 - K63

HÀ NỘI – 2023

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục tiêu

(a)

(b)

(c)

2. Nội dung

(a)

(b)

(c)

3. Đánh giá kết quả đạt được

(a)

(b)

(c)

Hà Nội, ngày 01 tháng 07 năm 2023

Giảng viên

ThS. NGUYỄN TUẤN DŨNG

Lời cảm ơn

Bài báo cáo lần này là kết quả của quá trình học tập và nghiên cứu làm đồ án 2 trong kì 20222 vừa rồi của bản thân em. Em xin gửi lời cảm ơn chân thành đến Thầy Nguyễn Tuấn Dũng, trong thời gian vừa rồi đã hướng dẫn, định hướng nghiên cứu cho đồ án lần này của em. Thầy đã truyền đạt cho em không chỉ những kiến thức chuyên nghành cần có mà còn là tinh thần làm việc nghiêm túc, những lời chỉ điểm cho những vấn đề thực tế. Đồ án 2 là một bài tập lớn bổ ích giúp bản thân em tiếp xúc với nhiều kiến thức hay, bổ ích và nó hỗ trợ nhiều cho việc định hướng việc làm của bản thân em về sau. Mặc dù em đã cố gắng hết sức nhưng chắc chắn trong đồ án này của em khó có thể tránh khỏi những thiếu sót và nhiều chỗ chưa chính xác, kính mong cô xem xét và góp ý để bài báo cáo này được hoàn thiện hơn.
Em xin chân thành cảm ơn !

Hà Nội, tháng 07 năm 2023

Sinh viên

Long

Lê Hữu Đức Long

Tóm tắt nội dung Báo cáo

Machine learning là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các kĩ thuật cho phép các hệ thống "học" tự động từ dữ liệu để giải quyết những vấn đề cụ thể. Trong thời gian gần trở lại đây, machine learning dần được phổ biến là được ứng dụng vào nhiều khía cạnh trong cuộc sống. Trong báo cáo đồ án lần này em xin trình bày về việc xử lý dữ liệu trước khi chạy mô hình và sử dụng mô hình "Logistic Regression" để áp dụng cho bài toán dự đoán tình trạng của chuyến bay có bị delay hay không. Sau đây là một số nội dung chính trong bài báo cáo lần này

1. Bài toán phân lớp và các chỉ số đánh giá mô hình phân lớp
2. Thuật toán hồi quy logistic (Logistic regression).
3. Thuật toán cây quyết định CART (Decision Tree)
4. Tiền xử lý dữ liệu.
5. Chạy mô hình và đánh giá mô hình.
6. Định hướng phát triển.

Hà Nội, tháng 07 năm 2023

Tác giả

Long

Lê Hữu Đức Long

Mục lục

Mở đầu

Với sự phát triển của nghành khoa học, nhất là cuộc cách mạng công nghiệp 4.0 dường như đã là thay đổi thế giới rất nhanh chóng trong vòng 5 năm đổ lại gần đây. Trong cuộc cách mạng này nổi bật lên các ngành như Bigdata, I.O.T (Internet Of Things - Internet kết nối vạn vật), AI (Trí tuệ nhân tạo) và chúng được ứng dụng rất nhiều trong các lĩnh vực kinh doanh, đời sống xã hội,... Machine Learning(học máy) là một nhánh nhỏ trong nghành AI(trí tuệ nhân tạo), nó là một lĩnh vực nghiên cứu thuật toán và áp dụng trên máy tính cho phép máy tính có thể học, cải thiện khả năng làm việc với bài toán của nó dựa trên tập dữ liệu dùng để huấn luyện (training data). Học máy được chia là hai loại chính là dự đoán (prediction) và phân loại (classification). Trong báo cáo đồ án lần này của em, em sẽ trình bày về các bước để xây dựng một mô hình Machine Learning (Máy học) đơn giản để phân loại tình trạng chuyến bay có bị đến trễ hay không và đánh giá độ hiệu quả của mô hình. Bây giờ chúng ta cùng bắt đầu vào việc tìm hiểu các lý thuyết cơ bản trong Học máy cùng với sử dụng các package có trong ngôn ngữ lập trình Python để thực hiện.

Chương 1

Đặt vấn đề bài toán.

Đường vận tải hàng không là một trong những loại hình vận tải quan trong đối với nền kinh tế hiện nay. Trong một ngày có thể có hàng nghìn chuyến bay cất cánh đến và đi để phục vụ nhiều nhu cầu khác nhau của khách hàng. Một trong những vấn đề xuất hiện là delay chuyến bay.

Delay chuyến bay là tình trạng giờ khởi hành hoặc giờ hạ cánh của chuyến bay bị trì hoãn muộn hơn so với thời gian được thông báo và ghi trên vé của hành khách. Việc delay có thể diễn ra từ 30 phút đến vài tiếng đồng hồ, một chuyến bay có tình trạng delay từ một đến nhiều lần hoặc tệ hơn là hãng sẽ hủy chuyến bay. Có rất nhiều nguyên nhân dẫn đến tình trạng trên. Đối với trạm không lưu, thì họ sẽ cố gắng làm giảm việc này mức thấp nhất và đối với những khách hàng tham gia chuyến bay thì họ muốn tránh việc đặt phải những chuyến bay bị delay.

Trong đồ án lần này, em dự đoán việc một chuyến bay có bị delay hay không bằng cách áp dụng mô hình hồi quy Logistic và cây quyết định (CART) trong việc phân loại và dự đoán nhằm hỗ trợ phần nào công việc quản lý các chuyến bay đối với trạm không lưu cũng như đặt vé máy bay đối với khách hàng khi sử dụng dịch vụ. Em lựa chọn hai mô hình này cho bài toán vì đây là những thuật toán phổ biến cơ bản nhưng cũng hiệu quả và sẽ làm bước đệm tốt cho

việc nghiên cứu và xây dựng các thuật toán học máy khác mạnh mẽ, nâng cao hơn.

Chương 2

Bài toán phân lớp và các chỉ số đánh giá mô hình phân lớp.

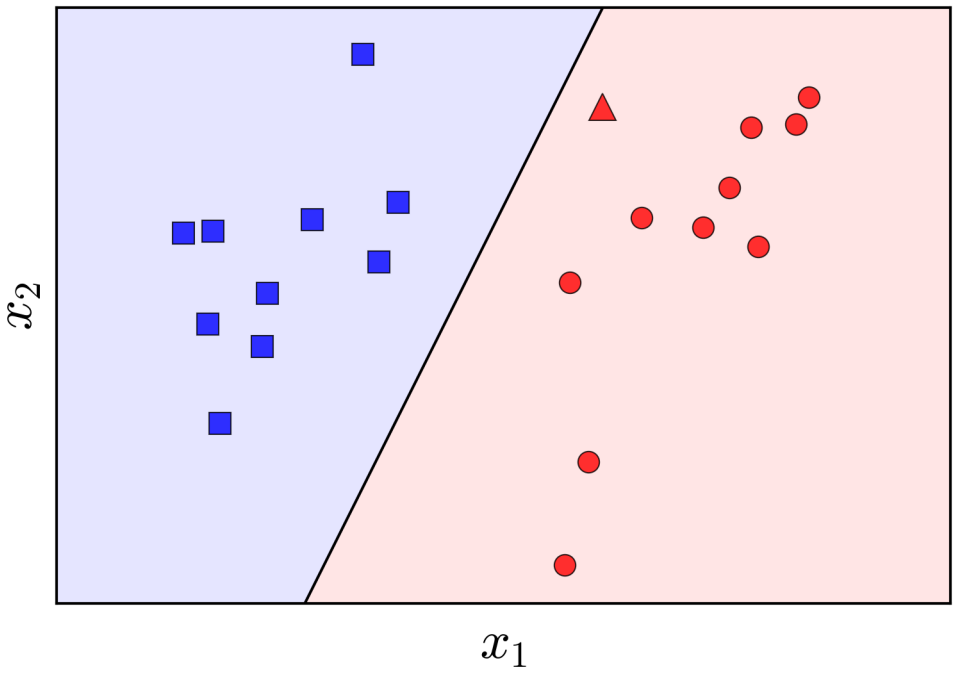
2.1 Bài toán phân lớp.

Trong học máy có giám sát, bài toán phân lớp là nơi chúng ta cố gắng xây dựng một mô hình để dự đoán và phân loại các điểm dữ liệu vào các lớp khác nhau dựa trên các đặc trưng của chúng.

Mô hình phân phân lớp được đào tạo với tập huấn luyện là những dữ liệu được gán và kiểm thử trên tập kiểm thử với các nhãn được gán tương đồng như trong tập huấn luyện. Đầu ra của bài toán phân lớp là việc dự đoán lớp hoặc nhãn của các điểm dữ liệu mới dựa trên mô hình đã được huấn luyện. Kết quả phân lớp có thể được biểu diễn dưới dạng các lớp nhãn hoặc các giá trị dự đoán.

Có nhiều bài toán phân lớp dữ liệu như phân lớp nhị phân (binary classification), phân lớp đa lớp (multiclass classification), phân lớp đa trị.

- Bài toán phân lớp nhị phân là bài toán gán nhãn dữ liệu cho đối tượng vào 1 trong 2 lớp khác nhau dựa vào việc dữ liệu đó có hay không có các đặc trưng (feature) của bộ phân lớp.
- Bài toán phân lớp đa lớp là quá trình phân lớp dữ liệu với số lượng lớp lớn



Hình 2.1: Phân lớp dữ liệu

hơn 2. Như vậy với từng dữ liệu phải xem xét và phân lớp chúng vào những lớp khác nhau chứ không phải là 2 lớp như bài toán phân lớp nhị phân. Và thực chất bài toán phân lớp nhị phân là 1 bài toán đặc biệt của phân lớp đa lớp.

Ứng dụng của bài toán này được sử dụng rất nhiều và rộng rãi trong thực tế ví dụ như phân loại các giao dịch trong kinh doanh, phân loại nhóm khách hàng, phân loại email, ...

2.2 Các chỉ số đánh giá mô hình phân lớp.

Trong học máy, mọi kết quả dự đoán đều có tỷ lệ sai lệch, để có thể biết được sự sai lệch là bao nhiêu thì ta cần phải đánh giá chúng.

Độ chính xác: accuracy

Độ chính xác (accuracy) là một phép đo thông dụng để đánh giá hiệu suất của mô hình phân loại. Nó được tính bằng tỉ lệ phần trăm giữa số lượng dự đoán chính xác và tổng số lượng điểm dữ liệu.

Độ chính xác được tính theo công thức:

Accuracy = (Số lượng dự đoán chính xác) / (Tổng số lượng điểm dữ liệu)

Chỉ số F1-score.

F1-score là một phép đo thống kê. Nó cung cấp một cách tổng hợp để đánh giá hiệu suất của mô hình dựa trên cả khả năng đúng dự đoán (precision) và khả năng bắt được tất cả các trường hợp dương tính (recall). F1-score được tính dựa trên công thức sau:

F1-score = 2 * (precision * recall) / (precision + recall)

Đường cong phân loại ROC.

Đường cong ROC trong hồi quy logistic được sử dụng để xác định giá trị ngưỡng tốt nhất để dự đoán liệu một quan sát mới là "thất bại" (0) hay "thành công" (1). Nếu bạn không quen thuộc với các đường cong ROC, bạn có thể cần nỗ lực để hiểu chúng. Đường cong này hiển thị hai tham số:

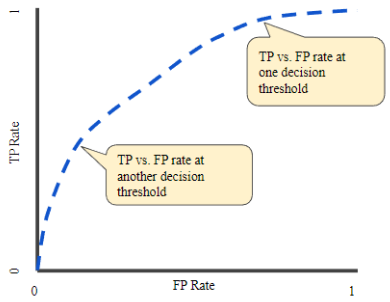
- Tỷ lệ dương tính thực - True positive rate (TPR)
- Tỷ lệ dương tính giả - False positive rate (FPR)

Trong đó:

$$\text{TPR} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{FPR} = \frac{\text{False Positive}}{\text{True Negative} + \text{False Negative}}$$

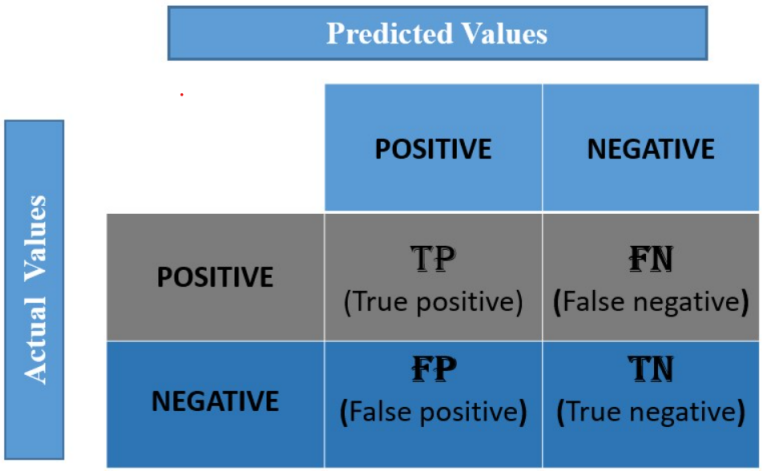
Đường cong ROC vẽ đồ thị TPR so với FPR ở các ngưỡng phân loại khác nhau. Việc giảm ngưỡng phân loại sẽ phân loại nhiều mục hơn, do đó tăng cả chỉ số FPR và TPR. Hình (2.2) cho thấy đường cong ROC điển hình.



Hình 2.2: Tỷ lệ TP so với FP ở các ngưỡng phân loại khác nhau.

Ma trận nhầm lẫn.

Ma trận nhầm lẫn (confusion matrix) là một công cụ để đánh giá hiệu suất của mô hình phân loại. Nó biểu thị sự khác biệt giữa các dự đoán của mô hình và nhãn thực tế. Ma trận nhầm lẫn như hình (2.3). Với P - Positive, N - Negative,



Hình 2.3: Ma trận nhầm lẫn

ta có thể hiểu là :

- TN (True Negative): Số lượng các điểm dữ liệu N được phân loại đúng vào lớp N (negative class).
- FP (False Positive): Số lượng các điểm dữ liệu N được phân loại sai vào lớp P (positive class).
- FN (False Negative): Số lượng các điểm dữ liệu P được phân loại sai vào lớp N.
- TP (True Positive): Số lượng các điểm dữ liệu P được phân loại đúng vào lớp P.

Ma trận nhầm lẫn cung cấp thông tin chi tiết về khả năng dự đoán đúng và sai của mô hình phân loại. Từ ma trận nhầm lẫn, có thể tính toán các chỉ số đánh giá như:

Accuracy - độ chính xác và F1-score.

$$accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$
$$F1 - score = 2 \times \frac{precision.recall}{precision + recall}$$

Precision là tỉ lệ tích cực (positive) dự đoán chính xác, được tính bằng số lượng tích cực thật chia cho tổng số lượng dự đoán là tích cực. Precision đo độ chính xác mô hình tốt như thế nào trong việc dự đoán một trường hợp cụ thể.

$$precision = \frac{TP}{TP + FP}$$

Recall là tỉ lệ tích cực thật, được tính bằng số lượng tích cực dự đoán đúng trên tổng số lượng tích cực.

$$recall = \frac{TP}{TP + FN}$$

2.3 Overfitting.

Overfitting trong học máy (machine learning) là tạo một mô hình (model) khớp với tập dữ liệu huấn luyện (training data) đến mức mô hình không đưa ra

dự đoán chính xác về dữ liệu mới.

Overfiting nó cũng có các dấu hiệu nhận biết để ta có thể điều chỉnh cho mô hình học máy. Sau đây là một số dấu hiệu mà ta có thể nhận ra:

- Hiệu suất tốt trên dữ liệu huấn luyện nhưng kém trên dữ liệu kiểm tra: Mô hình có thể đạt độ chính xác cao trên tập huấn luyện, nhưng khi áp dụng lên dữ liệu kiểm tra, hiệu suất giảm đáng kể.
- Quá khớp dữ liệu: Mô hình "nhớ" dữ liệu huấn luyện quá mức, thể hiện qua việc tạo ra quy tắc quá chi tiết và phức tạp, điều này dẫn đến khả năng dự đoán kém trên dữ liệu mới.
- Độ lỗi (loss) trên tập kiểm tra tăng lên: Trong quá trình huấn luyện, độ lỗi trên tập kiểm tra tăng lên sau một số lượt huấn luyện, trong khi độ lỗi trên tập huấn luyện tiếp tục giảm.

Đối với các trường hợp mô hình bị overfiting, các nhà phát triển đưa ra một số giải pháp sau để có thể đối phó với nó. Việc đối phó với overfitting đòi hỏi sự cân nhắc và thử nghiệm, và không có một phương pháp duy nhất phù hợp cho tất cả các tình huống. Tùy thuộc vào bộ dữ liệu và mô hình cụ thể, bạn cần điều chỉnh và tinh chỉnh các tham số và kỹ thuật để đạt được mô hình có hiệu suất tốt và khả năng tổng quát hóa cao. Dưới đây là một số cách mà em tham khảo được:

- Thu thập thêm dữ liệu: Cung cấp thêm dữ liệu huấn luyện để tăng tính đại diện của mô hình và giảm khả năng overfitting.
- Giảm độ phức tạp của mô hình: Giảm độ sâu của cây quyết định, giảm số lượng các nhánh hoặc giảm số lượng các biến đầu vào để làm mô hình đơn giản hơn.
- Sử dụng kỹ thuật regularization: Áp dụng kỹ thuật regularization như L1 hoặc L2 regularization để giới hạn độ lớn của trọng số trong mô hình.

- Sử dụng kỹ thuật cross-validation: Sử dụng phương pháp cross-validation để đánh giá hiệu suất của mô hình trên nhiều tập dữ liệu con khác nhau và lựa chọn mô hình có hiệu suất tốt nhất trên tập kiểm tra.
- Ensemble learning (kết hợp các mô hình học máy): Sử dụng phương pháp ensemble learning như trong thuật toán Random Forest hoặc thuật toán Gradient Boosting để kết hợp nhiều mô hình yếu lại với nhau và giảm khả năng overfitting.

Chương 3

Mô hình Logistic Regression (Hồi quy logistic)

3.1 Giới thiệu

Logistic Regression là một trong những thuật toán cơ bản của Machine Learning. Nó thuộc nhóm Supervised learning (học có giám sát). Logistic Regression là một thuật toán được sử dụng để phân loại nhị phân. Thuật toán này được áp dụng để dự đoán xác suất xảy ra của một biến phụ thuộc nhị phân dựa trên các biến độc lập. VD: Một người có sống sót sau tai nạn này hay không, Học sinh có vượt qua kỳ thi này hay không. Kết quả có thể là có hoặc không (2 đầu ra).

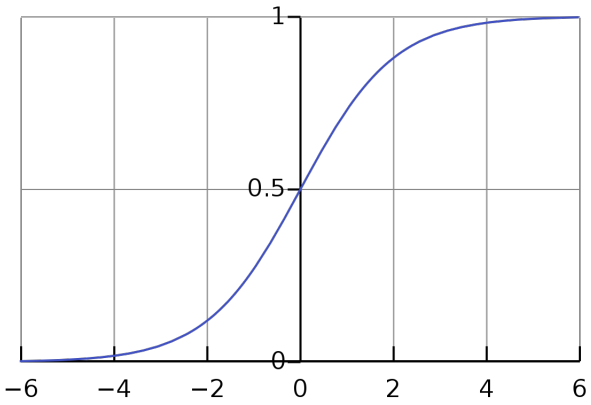
3.2 Hàm sigmoid

Cái tên "hồi quy logistic" bắt nguồn từ khái niệm về hàm logistic mà nó sử dụng. Hàm logistic thường được sử dụng còn được gọi là hàm sigmoid. Sigmoid là một hàm phi tuyến với đầu vào là các số thực và giá trị của hàm logistic này là các giá trị xác xuất được nằm trong khoảng từ 0 đến 1. Hàm sigmoid có dạng

sau:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

Đồ thị hàm số của hàm sigmoid:



Hình 3.1: Đồ thị hàm Sigmoid

Ta có thể dễ dàng chứng minh giá trị của hàm sigmoid nằm trong khoảng từ $[0, 1]$. Thật vậy:

$$\lim_{x \to +\infty} \sigma(x) = \lim_{x \to +\infty} \frac{1}{1 + e^{-x}} = 1 \tag{3.2}$$

và

$$\lim_{x \to -\infty} \sigma(x) = \lim_{x \to -\infty} \frac{1}{1 + e^{-x}} = 0 \tag{3.3}$$

Do đó hàm Sigmoid rất phù hợp để áp dụng vào dự báo xác suất ở các bài toán phân loại và cũng chính vì vậy ta thường sử dụng hàm Sigmiod cho hồi quy logistic.

3.3 Mô hình Logistic Regression

3.3.1 Lý thuyết toán học

Như đã biết, hồi quy logistic là một loại thuật toán học có giám sát tính toán mối quan hệ giữa các thuộc tính trong giá trị đầu vào (input) và giá trị dự

đoán (output) dựa trên hàm logistic(sigmoid). Không giống như các thuật toán hồi quy khác là dự đoán giá trị thực, hồi quy logistic dự đoán ra một kết quả nhị phân (1 hoặc 0, true hoặc false) dự vào giá trị đầu vào của nó. Trong quá trình xây dựng mô hình hồi quy logistic sẽ có chút giống với hồi quy tuyến tính (một thuật toán hồi quy cơ bản trong học máy). Các giá trị input có dạng:

$$f(x) = w_0 + w_1 x + ... + w_n x_n = w^T X \tag{3.4}$$

trong đó x là một giá trị quan sát. Với n giá trị quan sát, các giá trị này tạo thành một ma trận có kích thước là $n \times k$ với mỗi dòng là các giá trị quan sát thứ i, mỗi cột là 1 biến và n là số các giá trị quan sát được. Như vậy ta được một ma trận X như sau:

$$f(X) = \begin{pmatrix} X_{11} & \dots & X_{j1} & \dots & X_{k1} \\ X_{12} & \dots & X_{j1} & \dots & X_{k2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ X_{1n} & \dots & X_{jn} & \dots & X_{kn} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_k \end{pmatrix} = Xw \tag{3.5}$$

Đối với hồi quy tuyến tính, mục đích của thuật toán là ước tính các giá trị $w_0, w_1, ..., w_n$ và phù hợp nó với dữ liệu huấn luyện với hàm loss tối thiểu và dự đoán đầu ra y. Hồi quy logistic cũng làm điều tương tự, tuy vậy nó chạy kết quả thông qua một hàm non-linear (phi tuyến tính) là hàm sigmoid để tạo ra đầu ra là một xác suất P.

Do $f(x)$ có giá trị từ 0 đến 1 nên thay vì xét giá trị của $f(x)$ ta xét $\ln(\frac{f(x)}{1-f(x)})$ đối với mỗi một giá trị quan sát:

$$\ln(\frac{f(x)}{1-f(x)}) = w_0 + w_1 x + ... + w_n x_n = w^T X \tag{3.6}$$

Với mỗi xác suất $f(x) = P$, model sẽ đưa ra dự đoán theo công thức sau:

$$\begin{cases} 0 & , f(x) < 0.5 \\ 1 & , f(x) \geq 0.5 \end{cases} \tag{3.7}$$

3.3.2 Giải thuật

Xây dựng hàm mất mát

Trên thực tế, dữ liệu đầu vào sẽ có nhiều biến dữ liệu tương ứng với nhiều giá trị quan sát, mỗi một giá trị quan sát tương ứng với một điểm x. Nhiệm vụ của bài toán là phân loại chúng dựa trên giá trị xác suất (3.7). Điểm x sẽ thuộc lớp "1" là $\left[f(w^T x)\right]$ và thuộc lớp "0" sẽ là $\left[1 - f(w^T x\right])$. Với mỗi điểm dữ liệu trainning (đã biết dữ liệu y), ta có thể viết lại như sau:

$$P(y_i = 1 \mid x_i; w) = f(w^T x_i) \tag{3.8}$$

$$P(y_i = 0 \mid x_i; w) = 1 - f(w^T x_i) \tag{3.9}$$

Trong đó $P(y_i = 1 \mid x_i; w)$ chính là xác suất có điều kiện và được hiểu rằng xác suất xảy ra $y_i = 1$ khi biết các tham số w, x_i, tương tự với $y_i = 0$. Mục đích của ta là tìm các hệ số w sao cho $f(w^T x_i)$ có giá trị gần 1 nhất với các điểm dữ liệu thuộc lớp 1, ngược lại với những điểm thuộc lớp 0 thì giá trị $1 - f(w^T x_i)$ càng gần 0 càng tốt.

Đặt $z_i = f(w^T x_i)$, kết hợp công thức từ (3.8) và (3.9) ta được :

$$P(y_i \mid x_i; w) = z_i^{y_i}(1 - z_i)^{1-y_i} \tag{3.10}$$

Khi $y_i = 1$ vế phải (3.10) sẽ là $z_i = f(w^T x_i)$, còn khi $y_i = 0$ vế phải (3.10) sẽ trở thành $1 - z_i = 1 - f(w^T x_i)$. Như đã nêu ở trên, mục tiêu của chúng ta là muốn mô hình cho ra kết quả gần với dữ liệu đã cho nhất hay nói cách khác là làm cho xác suất đạt giá trị cao nhất.

Xét toàn bộ training set với $X \in \mathbb{R}^{(d+1)\times n}$ và $Y = [y_1, y_2, ..., y_N]^T$, chúng ta cần tìm w để biểu thức $P(y \mid X; w)$ đạt giá trị lớn nhất. Ở đây, X, y sẽ là các biến ngẫu nhiên. Khi đó bài toán của chúng ta là tìm tham số w để mô hình cho ra giá trị dự đoán gần nhất với dữ liệu đã có. Ta có thể hiểu như sau:

$$w = argmax_w P(Y \mid X; w) \tag{3.11}$$

Đây là một bài toán điển hình có tên "maximum likelihood estimation" (ước lượng hợp lý cực đại), hàm số argmax được gọi là *hàm ước lượng* do trong phạm vi đồ án 2 của em lần này nên em ko tìm hiểu sâu về bài toán trên.

Giả sử thêm rằng các dữ liệu được sinh ra một cách ngẫu nhiên độc lập với nhau, theo tính chất của xác suất độc lập ta có:

$$P(Y \mid X; w) = \prod_{i=1}^{N} P(y_i \mid x_i; w) = \prod_{i=1}^{N} z_i^{y_i}(1 - z_i)^{1-y_i} \tag{3.12}$$

Nhìn vào biểu thức ta có thể thấy rằng việc tối ưu trực tiếp hàm số này theo w không đơn giản chút nào bởi tích của N số nhỏ hơn 1 (do giá trị hàm số thuộc $[0, 1]$ như đã nêu ở phần trên) có thể dẫn tới sai số trong tính toán do tích là một số quá nhỏ. Thông thường, để giải quyết bài toán này ta thường lấy logarit cơ số e. Cách này sẽ biến phép nhân thành phép cộng để tránh việc giá trị số quá nhỏ. Sau đó lấy ngược dấu để được một hàm và coi nó là hàm mất mát. Khi đó, bài toán tìm giá trị lớn nhất ban đầu chuyển thành tìm giá trị nhỏ nhất của hàm mất mát:

$$\mathcal{L}(w) = -\log P(Y \mid X; w) = -\sum_{i=1}^{N}(y_i \log z_i + (1 - y_i)\log(1 - z_i)) \tag{3.13}$$

3.3.3 Tối ưu hàm mất mát

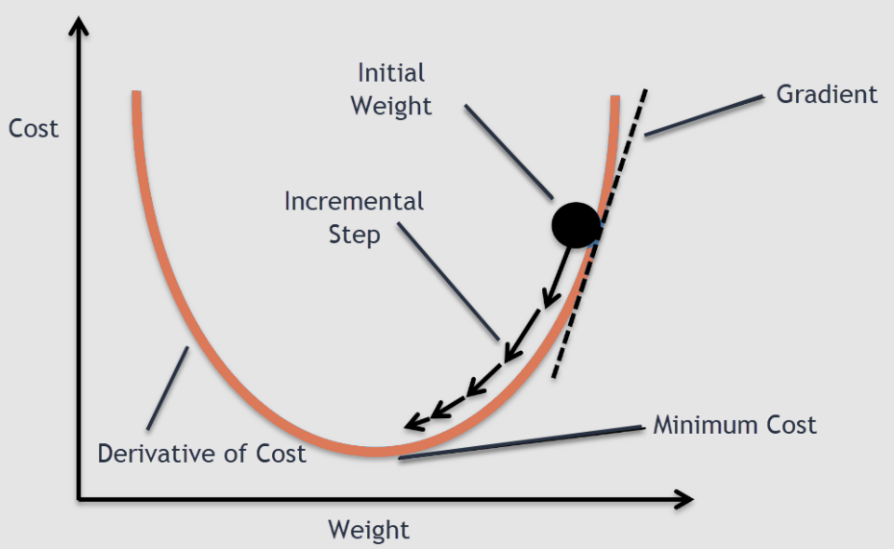
Gradient Descent

Trong Machine Learning nói riêng và Toán Tối Ưu nói chung, chúng ta thường xuyên phải tìm giá trị nhỏ nhất (hoặc đôi khi là lớn nhất) của một hàm số nào đó.Nhìn chung, việc tìm giá trị nhỏ nhất (gobal minimum) của các hàm mất mát trong học máy là rất phức tạp, thậm chí là bất khả thi. Thay vào đó, người ta thường cố gắng tìm các điểm giá trị nhỏ nhất cục bộ (local minimum) và ở một mức độ nào đó, coi đó là nghiệm cần tìm của bài toán.

Các điểm giá trị nhỏ nhất cục bộ là nghiệm của phương trình đạo hàm bằng 0. Nếu bằng một cách nào đó có thể tìm được toàn bộ (hữu hạn) các điểm

cực tiểu, ta chỉ cần thay từng điểm giá trị nhỏ nhất cục bộ đó vào hàm số rồi tìm điểm làm cho hàm có giá trị nhỏ nhất. Tuy nhiên, trong hầu hết các trường hợp, việc giải phương trình đạo hàm bằng 0 là bất khả thi. Nguyên nhân có thể đến từ sự phức tạp của dạng của đạo hàm, từ việc các điểm dữ liệu có số chiều lớn, hoặc từ việc có quá nhiều điểm dữ liệu.

Hướng tiếp cận phổ biến nhất là xuất phát từ một điểm mà chúng ta coi là gần với nghiệm của bài toán, sau đó dùng một phép toán lặp để tiến dần đến điểm cần tìm, tức đến khi đạo hàm gần với 0 (Hình 3.2). Gradient Descent dịch ra tiếng Việt là giảm dần độ dốc, nên hướng tiếp cận ở đây là chọn 1 nghiệm ngẫu nhiên cứ sau mỗi vòng lặp (hay epoch) thì cho nó tiến dần đến điểm cần tìm.



Hình 3.2: Mô phỏng thuật toán gradient descent

Công thức tổng quát chung:

$$\theta^{(\text{Kế tiếp})} = \theta - \eta \nabla_\theta \tag{3.14}$$

trong đó

- θ là tập các biến cần cập nhật,

- η là tốc độ học (learning rate),
- $\nabla\theta f(\theta)$ là Gradient của hàm mất mát theo tập θ.

Learning rate

Learning rate là một trong số những siêu tham số quan trọng nhất của mô hình. Có thể sẽ có các siêu tham số các bạn chưa nghe bao giờ hoặc chưa dùng bao giờ, nhưng learning rate, batch size, epochs, ... là các tham số các bạn bắt gặp hầu như trong hầu hết các bài toán Deep Learning trong quá trình huấn luyện.

Learning rate được hiểu là một phần tỷ lệ của một bước dịch chuyển trọng số mô hình được cập nhật theo các mini-batch truyền vào. Độ lớn của learning rate sẽ ảnh hưởng trực tiếp tới tốc độ hội tụ của hàm loss tới điểm cực trị toàn cục.

Thuật toán Gradient Descent minh họa:

Giả sử x_t là điểm ta tìm được sau vòng lặp thứ t. Ta cần một thuật toán để đưa x_t về càng gần $x*$ càng tốt.

Nếu đạo hàm của hàm số tại $x_t : f'(x_t) > 0$ thì x_t nằm về bên phải so với x^* (và ngược lại). Để điểm tiếp theo x_{t+1} gần với x^* hơn, chúng ta cần di chuyển x_t về phía bên trái, tức về phía âm. Nói các khác, chúng ta cần di chuyển ngược dấu với đạo hàm:

$$x_{t+1} = x_t + \Delta$$

Trong đó Δ là một đại lượng ngược dấu với đạo hàm $f'(x_t)$. x_t càng xa x^* về phía bên phải thì $f'(x_t)$ càng lớn hơn 0 (và ngược lại). Vậy, lượng di chuyển Δ, một cách trực quan nhất, là tỉ lệ thuận với $-f'(x_t)$. Hai nhận xét phía trên cho chúng ta một cách cập nhật đơn giản là:

$$x_{t+1} = x_t - \eta f'(x_t)$$

Trong đó η (đọc là eta) là một số dương là kí hiệu cho learning rate (tốc độ học).

Dấu trừ thể hiện việc chúng ta phải đi ngược với đạo hàm (Đây cũng chính là lý do phương pháp này được gọi là Gradient Descent - descent nghĩa là đi ngược). Các quan sát đơn giản phía trên, mặc dù không phải đúng cho tất cả các bài toán, là nền tảng cho rất nhiều phương pháp tối ưu nói chung và thuật toán Machine Learning nói riêng.

Stochastic Gradient Descent (SGD).

Đây là một biến thể của Gradient Descent.Trong thuật toán này, tại 1 thời điểm, ta chỉ tính đạo hàm của hàm mất mát dựa trên chỉ một điêm dữ liệu $\mathbf{x_i}$ rồi cập nhật θ dựa trên đạo hàm này. Việc này được thực hiện với từng điểm trên toàn bộ dữ liệu, sau đó lặp lại quá trình trên. Thuật toán rất đơn giản này trên thực tế lại làm việc rất hiệu quả. Mỗi lần duyệt một lượt qua tất cả các điểm trên toàn bộ dữ liệu người ta gọi là một epoch (i-póc). Với $gradient - descent$ thông thường thì mỗi epoch ứng với 1 lần cập nhật θ, với SGD thì mỗi epoch ứng với N lần cập nhật θ với N là số điểm dữ liệu. Nhìn vào một mặt, việc cập nhật từng điểm một như thế này có thể làm giảm đi tốc độ thực hiện 1 epoch. Nhưng nhìn vào một mặt khác, SGD chỉ yêu cầu một lượng epoch rất nhỏ (thường là 10 cho lần đầu tiên, sau đó khi có dữ liệu mới thì chỉ cần chạy dưới một epoch là đã có nghiệm tốt). Vì vậy SGD phù hợp với các bài toán có lượng cơ sở dữ liệu lớn và các bài toán yêu cầu mô hình thay đổi liên tục (online learning).

Tối ưu hàm mất mát

Ta đã xây dựng hàm mất mát cho thuật toán Logistic regression:

$$\mathcal{L}(w) = -\log P(Y \mid X; w) = -\sum_{i=1}^{N}(y_i \log z_i + (1 - y_i)\log(1 - z_i))$$

Để tối ưu hàm mất mát này, ta sẽ sử dụng một biến thể của Gradient Descent là Stochastic Gradient Descent.

Tối ưu trên điểm dữ liệu:

Một điểm cần lưu ý đó là: sau mỗi epoch, chúng ta cần xáo trộn thứ tự của các dữ liệu để đảm bảo tính ngẫu nhiên. Việc này cũng ảnh hưởng tới hiệu năng của SGD. Một cách toán học, quy tắc cập nhật của SGD là:

$$\theta = \theta - \eta \nabla_\theta J\left(\theta; \mathbf{x}_i; \mathbf{y}_i\right)$$

trong đó $J\left(\theta; \mathbf{x}_i; \mathbf{y}_i\right)$ là hàm mất mát với chỉ 1 cặp điểm dữ liệu (input, label) là $(\mathbf{x}_i, \mathbf{y}_i)$.

Khi đó hàm mất mát (1.13) với 1 điểm dữ liệu (x_i, y_i) là:

$$J\left(\mathbf{w}; \mathbf{x}_i, y_i\right) = -\left(y_i \log z_i + (1 - y_i) \log\left(1 - z_i\right)\right) \tag{3.15}$$

Đạo hàm:

$$\frac{\partial J\left(\mathbf{w}; \mathbf{x}_i, y_i\right)}{\partial \mathbf{w}} = -\left(\frac{y_i}{z_i} - \frac{1 - y_i}{1 - z_i}\right) \frac{\partial z_i}{\partial \mathbf{w}} \tag{3.16}$$

$$= \frac{z_i - y_i}{z_i\left(1 - z_i\right)} \frac{\partial z_i}{\partial \mathbf{w}} \tag{3.17}$$

Để cho biểu thức này trở nên gọn và đẹp hơn, chúng ta sẽ tìm hàm $z = f\left(\mathbf{w}^T \mathbf{x}\right)$ sao cho mẫu số bị triệt tiêu. Nếu đặt $s = \mathbf{w}^T \mathbf{x}$, chúng ta sẽ có:

$$\frac{\partial z_i}{\partial \mathbf{w}} = \frac{\partial z_i}{\partial s} \frac{\partial s}{\partial \mathbf{w}} = \frac{\partial z_i}{\partial s} \mathbf{x} \tag{3.18}$$

Một cách trực quan nhất, ta sẽ tìm hàm số $z = f(s)$ để triệt tiêu mẫu số trong biểu thức (3.17) thỏa mãn:

$$\frac{\partial z}{\partial s} = z(1 - z)$$

Phương trình trên tương đương với:

$$\begin{aligned}
\tfrac{\partial z}{z(1-z)} &= \partial s \\
\Leftrightarrow \left(\tfrac{1}{z} + \tfrac{1}{1-z}\right) \partial z &= \partial s \\
\Leftrightarrow \log z - \log(1 - z) &= s \\
\Leftrightarrow \quad \log \tfrac{z}{1-z} &= s \\
\Leftrightarrow \quad \tfrac{z}{1-z} &= e^s \\
\Leftrightarrow \quad z &= e^s(1 - z) \\
\Leftrightarrow \quad z = \tfrac{e^s}{1+e^s} &= \tfrac{1}{1+e^{-s}} = \sigma(s)
\end{aligned}$$

Như vậy hàm số $z = f(s)$ cần tìm là $\sigma(s) = \dfrac{1}{1+e^s}$, cũng chính là hàm Sigmoid.

Đến đây, ta có:

$$\frac{\partial J\left(\mathbf{w}; \mathbf{x}_i, y_i\right)}{\partial \mathbf{w}} = \left(z_i - y_i\right)\mathbf{x}_i \tag{3.19}$$

Áp dụng thuật toán SGD cho logistic regression ta được:

$$\mathbf{w} = \mathbf{w} + \eta\left(y_i - z_i\right)\mathbf{x}_i \tag{3.20}$$

Ta đã tối ưu được hàm mất mát đối với mô hình Logistic Regression, trong phần tiếp theo sẽ áp dụng mô hình với bộ dữ liệu cụ thể giải quyết bài toán đã nêu ở phần đầu.

3.4 Mô tả cách hoạt động của thuật toán.

1. Sau khi có bộ dữ liệu bao gồm biến phụ thuộc nhị phân và các biến độc lập, ta sử dụng hàm logistic (hay hàm sigmoid) để biểu diễn xác suất xảy ra của biến phụ thuộc. Hàm logistic chuyển đổi giá trị đầu vào thành một giá trị trong khoảng từ 0 đến 1, thể hiện xác suất xảy ra.

2. Quá trình huấn luyện của thuật toán bao gồm tìm kiếm các trọng số tối ưu để tối thiểu hóa hàm mất mát (loss function). Mục tiêu là điều chỉnh các trọng số sao cho mô hình dự đoán xác suất xảy ra gần với giá trị thực tế nhất. Để tìm ra các trọng số tối ưu ta sử dụng phương pháp tối ưu hóa stochastic gradient descent. Thuật toán sẽ điều chỉnh các trọng số dựa trên đạo hàm của hàm mất mát để tìm điểm cực tiểu.

Chương 4

Mô hình Cây quyết định (CART)

4.1 Giới thiệu mô hình cây quyết định.

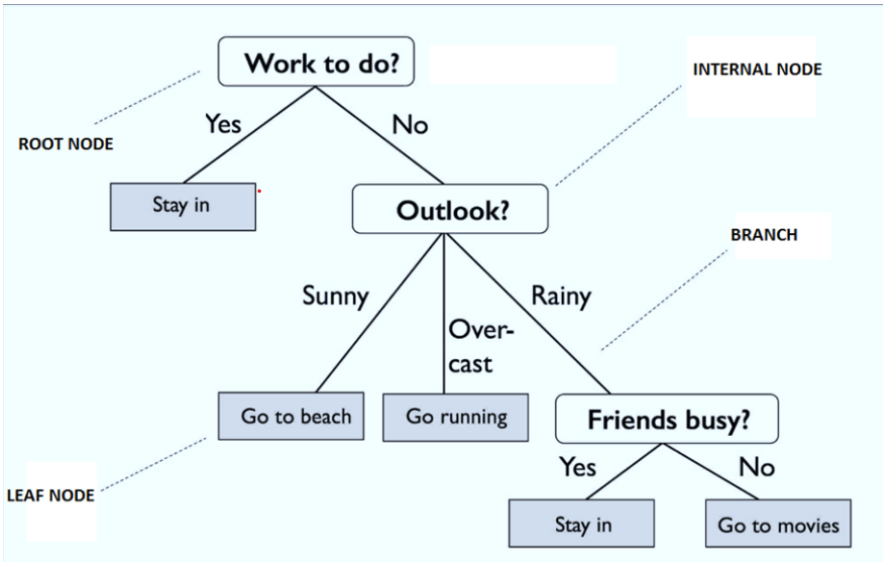
Cây quyết định (Decision Tree) là một thuật toán học máy được sử dụng cho bài toán phân loại và dự đoán. Nó là một mô hình dự đoán được biểu diễn dưới dạng cây có cấu trúc phân cấp, trong đó mỗi nút trong cây đại diện cho một điều kiện hoặc thuộc tính và các nhánh của cây đại diện cho các quyết định hoặc kết quả.

Cây quyết định bắt đầu với một nút gốc và tại mỗi nút, thuật toán đánh giá các thuộc tính của dữ liệu để chọn thuộc tính tốt nhất để phân chia các mẫu dữ liệu thành các nhóm. Quá trình này được tiếp tục cho đến khi các mẫu dữ liệu ở mỗi nhánh được phân loại vào cùng một lớp hoặc không thể phân loại thêm được.

Một cây quyết định bao gồm (Hình 4.1):

- "Root node": điểm ngọn chứa giá trị của biến đầu tiên được dùng để phân nhánh.
- "Internal node": các điểm bên trong thân cây là các biến chứa các giá trị dữ liệu được dùng để xét cho các phân nhánh tiếp theo.
- "Leaf node": là các lá cây chứa giá trị của biến phân loại sau cùng.

- "Branch" là quy luật phân nhánh, nói đơn giản là mối quan hệ giữa giá trị của biến độc lập (Internal node) và giá trị của biến mục tiêu (Leaf node).



Hình 4.1: Mô hình cây quyết định

4.2 Mô hình cây quyết định với thuật toán CART.

Thuật toán CART (Classification and Regression Trees) là một thuật toán quyết định dựa trên cây được sử dụng cho bài toán phân loại và hồi quy. CART sử dụng cách tiếp cận phân chia đệ quy để xây dựng cây quyết định, trong đó mỗi nút trong cây đại diện cho một điều kiện hoặc thuộc tính và các nhánh của cây đại diện cho các quyết định hoặc kết quả thông qua chỉ số 'Gini'.

4.2.1 Chỉ số Gini

Đây là một chỉ số dùng để đánh giá xem việc phân chia ở node điều kiện có tốt hay không thông qua việc tìm kiếm các thuộc tính đồng nhất và thuật toán CART sử dụng chỉ số Gini để tạo phân tách nhị phân cho một phân vùng dữ liệu hoặc tập hợp các bộ dữ liệu đào tạo.

Cho một tập dữ liệu D, một tập các nhãn C_i (i>=1 và $i <= m$ với m là số nhãn), trong đó:

C_i, D : là tất cả các bộ dữ liệu có nhãn lớp C_i trong D.

$|D|$: là tổng số bộ dữ liệu của tập dữ liệu D.

$|C_i, D|$: là tổng số bộ dữ liệu của tập dữ liệu D có nhãn lớp C_i.

Với các giả định trên thì chỉ số $Gini$ được định nghĩa như sau:

$$Gini(D) = 1 - \sum_{i=1}^{m} (p_i)^2 \tag{4.1}$$

Ở đây p_i là xác suất để một bộ bất kì trong D thuộc về nhãn lớp C_i và được tính bởi công thức $p_i = \frac{|C_{i,D}|}{|D|}$; m là tổng số nhãn lớp.

Chỉ số $Gini$ thường sẽ được tính toán dựa trên giả định một tập dữ liệu D được phân chia nhị phân thành hai tập con. Đầu tiên xét trường hợp thuộc tính A bất kỳ trong D có kiểu dữ liệu rời rạc, khi dùng phép chiếu thu được $v = a_1, a_2, ..., a_v$ giá trị khách nhau. Để xác định điểm chi tốt nhất của A, kiểm tra tất cả tập con có thể tạo được từ giá trị phân biệt trên, mỗi tập con tạm gọi là S_A là một điều kiện kiểm tra nhị phân dạng $A \in S_A$. Như vật với v giá trị khác nhau ta sẽ có $2^v - 2$ tập con, trong đó tập rỗng và tập toàn phần $v = a_1, a_2, ..., a_v$ sẽ không được xét đến. Như vậy tiến hành lặp qua tất cả các tập con này, mỗi lần lặp sẽ phân chia tập giá trị v thành hai tập con v_1 và v_2 riêng biệt thỏa mãn điều điện rời rạc toàn phần (hội v_1 và v_2 chính là tập v và phần giao tập rỗng). Với hai tập con v_1 và v_2 này tương ứng tập con D cũng được phân chia thành hai tập con D_1 (các bộ phận có giá trị thuộc tính $A \in v_1$) và D_2 (các bộ có giá trị thuộc tính $A \in v_2$), $Gini(D)$ theo thuộc tính A ($Ginh_A(D)$) sẽ được tính như sau:

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \tag{4.2}$$

Ta sẽ chọn chỉ số $Gini$ nhỏ nhất với mong muốn sau khi phân chia sẽ làm giảm tính không trong suốt của tập D nhiều nhất (do giá trị Gini càng cao, tức là tập dữ liệu càng không trong suốt và chứa nhiều lớp khác nhau.). Đối với các giá trị liên tục có một lưu ý là đầu tiên phải sắp xếp các giá trị này, sau đó tất cả các giá trị cũng sẽ được tính toán chỉ số $Gini$ và chọn ra giá trị có chỉ số $Gini$

là nhỏ nhất. Mục tiêu của chỉ số *Gini* là làm giảm tính không trong suốt của dữ liệu càng nhiều càng tốt, điều này được thể hiện qua công thức sau:

$$\Delta gini(A) = Gini(D) - Gini_A(D) \tag{4.3}$$

Ở đây, $Gini(D)$ là một số cố định, chính vì mục đích chọn điểm chia sao cho $\Delta gini(A)$ là lớn nhất nên bắt buộc chọn thuộc tính A sao cho $Gini_A(D)$ là nhỏ nhất.

4.2.2 Overfiting của mô hình cây quyết định.

Đối với mô hình cây quyết định, thường sẽ hay xảy ra trong mô hình cây quyết định do đây là một mô hình phi tham số. Trong đồ án này của em, em sử dụng phương pháp thứ 2 (giảm độ phức tạp của mô hình) để giảm vấn đề overfitting cho mô hình. Điều này em sẽ trình bày cụ thể ở chương 5.

Chương 5

Ứng dụng mô hình.

Trong phần này, em sử dụng các thư viện trong ngôn ngữ Python để thực hiện việc tiền xử lý dữ liệu và xây dựng mô hình, như:

- Pandas
- Numpy
- Math
- Seaborn
- Matplotlib
- Sklearn

5.1 Giới thiệu tập dữ liệu.

Bộ dữ liệu của em được lấy từ Kaggle và nó chứa thông tin của các chuyến bay trong năm 2022 của Mĩ. Sau đây là các trường dữ liệu có trong bộ data và em đã phân loại chúng theo kiểu dữ liệu:

- Kiểu dữ liệu int64:
  - Year : Năm thực hiện chuyến bay.

- Quarter: Quý thực hiện chuyến bay.

- Month: Tháng thực hiện chuyến bay.

- DayofMonth: Ngày trong tháng thực hiện chuyến bay.

- DayOfWeek: Ngày trong tuần thực hiện chuyến bay.

- `DOT_ID_Marketing_Airline`: Số nhận dạng do US DOT chỉ định để xác định một hãng hàng không.

- `Flight_Number_Marketing_Airline`: Số hiệu chuyến bay.

- `DOT_ID_Operating_Airline`: Số nhận dạng do DOT Hoa Kỳ chỉ định để xác định một hãng hàng không duy nhất.

- `Flight_Number_Operating_Airline`: Số hiệu chuyến bay.

- OriginAirportID : ID sân bay xuất phát. Số nhận dạng do US DOT chỉ định để xác định một sân bay duy nhất.

- OriginAirportSeqID: ID sân bay xuất phát thay đổi theo thời gian. Số nhận dạng do US DOT chỉ định để xác định một sân bay duy nhất tại một thời điểm nhất định.

- OriginCityMarketID: ID thành phố có sân bay xuất phát.

- OriginStateFips: ID bang có sân bay xuất phát.

- OriginWac: Mã vùng thế giới của sân bay xuất phát.

- DestAirportID: ID sân bay đến.

- DestAirportSeqID: ID sân bay đến thay đổi theo thời gian.

- DestCityMarketID: ID thành phố có sân bay đến.

- DestStateFips: ID bang có sân bay đến .

- DestWac: Mã vung thế giới của sân bay đến.

- CRSArrTime: Thời gian đến theo CSR.

- DistanceGroup: khoảng cách của các sân bay trên 250 dặm.

- DivAirportLandings: Số lần hạ cánh tại sân bay chuyển hướng.

- Kiểu dữ liệu float64

  - DepTime: Thời gian khởi hành thực tế.

- DepDelayMinutes: Chênh lệch số phút giữa thời gian khởi hành theo lịch trình và thực tế. Các chuyến khởi hành sớm được đặt thành 0.
- DepDelay: Chênh lệch số phút giữa thời gian khởi hành theo lịch trình và thực tế. Khởi hành sớm cho thấy số âm.
- ArrTime: Thời gian bay thực tế. - ArrDelayMinutes: Sự khác biệt về số phút giữa thời gian đến theo lịch trình và thực tế. Số lượt đến sớm được đặt thành 0.
- AirTime: thời gian bay.
- CRSElapsedTime: Thời gian bay trong suốt quá trình bay theo CRS.
- ActualElapsedTime: Thời gian bay của chuyến bay trong suốt quá trình bay.
- Distance: Khoảng cách giữa các sân bay.
- DepDel15: Thời gian trễ trên 15p (1 hoặc 0).
- DepartureDelayGroups: Khoảng thời gian trì hoãn.
- WheelsOff: Thời gian bánh xe dừng lại.
- WheelsOn: Thời gian bánh xe bắt đầu lăn.
- TaxiIn: Thời gian xe taxi đến.
- ArrDelay: Sự khác biệt về số phút giữa thời gian đến theo lịch trình và thực tế. Những người đến sớm hiển thị số âm.
- ArrDel15: Độ trễ trên 15p (1 hoặc 0).
- ArrivalDelayGroups: Khoảng thời gian đến trễ (trên 15p)

- Kiểu object
  - Airline : Tên hãng hàng không.
  - Origin: Tên sân bay xuất phát.
  - Dest: Tên sân bay đến.
  - `Marketing_Airline_Network`: Mã nhà cung cấp dịch vụ tiếp thị duy nhất.
  - `Operated_or_Branded_Code_Share_Partners`: Báo cáo Đối tác chia sẻ mã thương hiệu hoặc nhà cung cấp dịch vụ vận hành.

- `IATA_Code_Marketing_Airline`: Mã được chỉ định bởi IATA và thường được sử dụng để xác định nhà cung cấp dịch vụ.
- `Operating_Airline`: Mã nhà cung cấp dịch vụ duy nhất.
- `IATA_Code_Operating_Airline`: Mã do IATA chỉ định và thường được sử dụng để xác định nhà cung cấp dịch vụ.
- `Tail_Number`: Số hiệu đuôi chuyến bay.
- OriginCityName: Tên thành phố có sân bây xuất phát.
- OriginState: Mã tiểu bang có sân bay xuất phát.
- OriginStateName: Tên tiểu bang có sân bay xuất phát.
- DestCityName: Tên thành phố có sân bay đến.
- DestState: Mã tiểu bang có sân bay đến.
- DestStateName: Tên tiêu bang có sân bay đến.
- DepTimeBlk: Khối thời gian khởi hành theo CRS.
- ArrTimeBlk: Khối thời gian đến theo CRS.

- Kiểu Bool
  - Cancelled: Chỉ báo chuyến bay bị hủy (1 hoặc 0).
  - Diverted: Chỉ báo chuyến bay chuyển hướng (1 hoặc 0).

5.2 Xử lý bộ dữ liệu.

Trong Học máy, công đoạn tiền xử lý dữ liệu luôn là một trong những công đoạn quan trọng để có thể sử dụng bộ dữ liệu này cho việc huấn luyện mô hình. Các kỹ thuật tiền xử lý dữ liệu phổ biến hiện nay bao gồm: xử lý dữ liệu bị khuyết (missing data), mã hóa các biến nhóm (encoding categorical variables), chuẩn hóa dữ liệu (standardizing data), co giãn dữ liệu (scaling data),...Đối với bộ dữ liệu trên ta cũng sẽ tiến hành vài bước để làm sạch bộ dữ liệu và biến đổi trước khi chạy mô hình Logistic regression.

FlightDate	Airline	Origin	Dest	Cancelled	Diverted	CRSDepTime	DepTime	DepDelayMinutes	DepDelay	ArrTime	...	WheelsOff	WheelsOn	TaxiIn	CRS.
2022-04-04	Commutair Aka Champlain Enterprises, Inc.	GJT	DEN	False	False	1133	1123.0	0.0	-10.0	1228.0	...	1140.0	1220.0	8.0	
2022-04-04	Commutair Aka Champlain Enterprises, Inc.	HRL	IAH	False	False	732	728.0	0.0	-4.0	848.0	...	744.0	839.0	9.0	
2022-04-04	Commutair Aka Champlain Enterprises, Inc.	DRO	DEN	False	False	1529	1514.0	0.0	-15.0	1636.0	...	1535.0	1622.0	14.0	
2022-04-04	Commutair Aka Champlain Enterprises, Inc.	IAH	GPT	False	False	1435	1430.0	0.0	-5.0	1547.0	...	1446.0	1543.0	4.0	
2022-04-04	Commutair Aka Champlain Enterprises, Inc.	DRO	DEN	False	False	1135	1135.0	0.0	0.0	1251.0	...	1154.0	1243.0	8.0	

5 rows × 60 columns

Hình 5.1: Bộ dữ liệu

5.2.1 Kiểm tra thông tin về bộ dữ liệu.

Với Pandas, ta dễ dàng đọc được bộ dữ liệu với câu lệnh `pd.read.csv('\path')`. Ta sẽ đặt tên cho bộ DataFrame này là `Flight_delay`, dưới đây là bộ dữ liệu sau khi được lấy ra: Ta sẽ kiểm tra qua thông tin về bộ dữ liệu:

`Flight_delay.info()`, kết quả được như sau:

```
<class 'pandas.core.frame.DataFrame'>
Index: 4078318 entries, 2022-04-04 to 2022-03-07
Data columns (total 60 columns):
 #    Column                                    Dtype
---   ------                                    -----
 0    Airline                                   object
 1    Origin                                    object
 2    Dest                                      object
 3    Cancelled                                 bool
 4    Diverted                                  bool
 5    CRSDepTime                                int64
 6    DepTime                                   float64
 7    DepDelayMinutes                           float64
 8    DepDelay                                  float64
 9    ArrTime                                   float64
```

```
10  ArrDelayMinutes                          float64
11  AirTime                                  float64
12  CRSElapsedTime                           float64
13  ActualElapsedTime                        float64
14  Distance                                 float64
15  Year                                     int64
16  Quarter                                  int64
17  Month                                    int64
18  DayofMonth                               int64
19  DayOfWeek                                int64
20  Marketing_Airline_Network                object
21  Operated_or_Branded_Code_Share_Partners  object
22  DOT_ID_Marketing_Airline                 int64
23  IATA_Code_Marketing_Airline              object
24  Flight_Number_Marketing_Airline          int64
25  Operating_Airline                        object
26  DOT_ID_Operating_Airline                 int64
27  IATA_Code_Operating_Airline              object
28  Tail_Number                              object
29  Flight_Number_Operating_Airline          int64
30  OriginAirportID                          int64
31  OriginAirportSeqID                       int64
32  OriginCityMarketID                       int64
33  OriginCityName                           object
34  OriginState                              object
35  OriginStateFips                          int64
36  OriginStateName                          object
37  OriginWac                                int64
38  DestAirportID                            int64
39  DestAirportSeqID                         int64
40  DestCityMarketID                         int64
41  DestCityName                             object
42  DestState                                object
43  DestStateFips                            int64
44  DestStateName                            object
45  DestWac                                  int64
46  DepDel15                                 float64
47  DepartureDelayGroups                     float64
48  DepTimeBlk                               object
```

```
 49  TaxiOut                                    float64
 50  WheelsOff                                  float64
 51  WheelsOn                                   float64
 52  TaxiIn                                     float64
 53  CRSArrTime                                 int64
 54  ArrDelay                                   float64
 55  ArrDel15                                   float64
 56  ArrivalDelayGroups                         float64
 57  ArrTimeBlk                                 object
 58  DistanceGroup                              int64
 59  DivAirportLandings                         int64
dtypes: bool(2), float64(18), int64(23), object(17)
memory usage: 1.8+ GB
```

Ta có thể thấy bộ dữ liệu này có hơn 4 triệu bản ghi, 60 trường dữ liệu và có các 4 loại kiểu dữ liệu được lưu trữ là bool, float64, int64, object. Đối với bộ dữ liệu, thông qua việc mô tả ở phần trên ta có thể thấy nó có khá nhiều các biến (trường) dữ liệu là các mã định danh (các số hiệu, ID) được thu thập, các biến dữ liệu này dường như không có ý nghĩa trong việc huấn luyện mô hình. Bởi vậy, em sẽ xóa các biến dữ liệu:

```
ID_col = ['TaxiIn','DOT_ID_Marketing_Airline',
      'Flight_Number_Marketing_Airline',
     'DOT_ID_Operating_Airline',
     'Flight_Number_Operating_Airline',
     'OriginAirportID',
     'OriginAirportSeqID',
     'OriginCityMarketID',
     'OriginStateFips',
     'OriginWac',
     'DestAirportID',
     'DestAirportSeqID',
     'DestCityMarketID',
     'DestStateFips',
     'DestWac',
     'IATA_Code_Marketing_Airline',
     'Operating_Airline',
     'IATA_Code_Operating_Airline',
```

```
    'Tail_Number',
    'DestState',
    'OriginState']
df = Flight_delay.drop(ID_col,axis=1)
```

Sau khi loại bỏ các biến dữ liệu định danh không mang nhiều ý nghĩa, ta bắt đầu đánh giá lại tổng quan bộ dữ liệu và kiểm tra các 'missing value':

```
print("Kich thuoc:")
print("Rows : ", df.shape[0])
print("Columns : ", df.shape[1])
# kiem tra o du lieu trong trong data
print("-"*75,"\n", "Du lieu trong: ","\n",
df.isnull().sum().sort_values(ascending=False))
```

kết quả được như sau:

```
    Kich thuoc:
Rows :  4078318
Columns :  39
---------------------------------------------------------------------------

 Du lieu trong:
 AirTime                                 133402
ActualElapsedTime                        133402
ArrivalDelayGroups                       133402
ArrDel15                                 133402
ArrDelay                                 133402
ArrDelayMinutes                          133402
WheelsOn                                 124242
ArrTime                                  124239
WheelsOff                                122666
TaxiOut                                  122666
DepDelayMinutes                          120495
DepartureDelayGroups                     120495
DepDel15                                 120495
DepDelay                                 120495
DepTime                                  120433
DepTimeBlk                                    0
DestStateName                                 0
```

```
Airline                                          0
CRSArrTime                                       0
OriginStateName                                  0
ArrTimeBlk                                       0
DistanceGroup                                    0
DestCityName                                     0
DayOfWeek                                        0
OriginCityName                                   0
Operated_or_Branded_Code_Share_Partners          0
Marketing_Airline_Network                        0
Origin                                           0
DayofMonth                                       0
Month                                            0
Quarter                                          0
Year                                             0
Distance                                         0
CRSElapsedTime                                   0
CRSDepTime                                       0
Diverted                                         0
Cancelled                                        0
Dest                                             0
DivAirportLandings                               0
dtype: int64
```

Như vậy, trong bộ dữ liệu tồn tại các dữ liệu trống nhưng trước khi xử lý chúng ta cần xử lý các giá trị âm và các biến có kiểu dữ liệu object. Trong bộ dữ liệu tồn tại 3 biến dữ liệu có các giá trị âm: 'DepDelay', 'DepartureDelayGroups', 'DepDelayMinutes', ta sẽ chuyển đổi các giá trị âm về 0 còn giá trị dương về 1. Bên cạnh đó, ta có thể thấy ý nghĩa của 2 biến dữ liệu 'DepDelayMinutes' và 'DepDel15' bị trùng ý nghĩa với 'DepDelay' nên ta cũng sẽ tiến hành xóa chúng.

```
def convert(k):
if k <= 0.0:
    return 0
else:
    return 1
df["DepDelay"] = df["DepDelay"].apply(lambda x: convert(x));
df["DepartureDelayGroups"] = df["DepartureDelayGroups"].apply(lambda x
```

```
    : convert(x));
df = df.drop(columns=['DepDelayMinutes','DepDel15'], axis=1)
```

Tiếp theo, ta cần mã hóa các biến có kiểu dữ liệu object. Với Pandas, ta dễ dàng mã hóa chúng nhưng cần phải chuyển chúng thành kiểu category.

```
df['Airline'] = df['Airline'].astype('category')
df['Airline'] = df['Airline'].cat.codes
#################
df['Origin'] = df['Origin'].astype('category')
df['Origin'] = df['Origin'].cat.codes
###############
df['Dest'] = df['Dest'].astype('category')
df['Dest'] = df['Dest'].cat.codes
##############
df['Marketing_Airline_Network'] = df['Marketing_Airline_Network'].
    astype('category')
df['Marketing_Airline_Network'] = df['Marketing_Airline_Network'].cat.
    codes
##############
df['Operated_or_Branded_Code_Share_Partners'] = df['
    Operated_or_Branded_Code_Share_Partners'].astype('category')
df['Operated_or_Branded_Code_Share_Partners'] = df['
    Operated_or_Branded_Code_Share_Partners'].cat.codes
##############
df['OriginCityName'] = df['OriginCityName'].astype('category')
df['OriginCityName'] = df['OriginCityName'].cat.codes
###############
df['OriginStateName'] = df['OriginStateName'].astype('category')
df['OriginStateName'] = df['OriginStateName'].cat.codes
###############
df['DestCityName'] = df['DestCityName'].astype('category')
df['DestCityName'] = df['DestCityName'].cat.codes
##############
df['DestStateName'] = df['DestStateName'].astype('category')
df['DestStateName'] = df['DestStateName'].cat.codes
##############
df['ArrTimeBlk'] = df['ArrTimeBlk'].astype('category')
df['ArrTimeBlk'] = df['ArrTimeBlk'].cat.codes
```

```
##############
df['Cancelled'] = df['Cancelled'].astype('category')
df['Cancelled'] = df['Cancelled'].cat.codes
######################
df['Diverted'] = df['Diverted'].astype('category')
df['Diverted'] = df['Diverted'].cat.codes
###############
df['DepTimeBlk'] = df['DepTimeBlk'].astype('category')
df['DepTimeBlk'] = df['DepTimeBlk'].cat.codes
```

Sau khi đã mã hóa các biến object và xử lý các biến có giá trị âm, ta bắt đầu xử lý các biến bị thiếu bằng cách thêm vào đó các giá trị trung bình.

```
df.DepTime.fillna(df.DepTime.mean(), inplace=True)
df.ArrTime.fillna(df.ArrTime.mean(), inplace=True)
df.ArrDelayMinutes.fillna(df.ArrDelayMinutes.mean(), inplace=True)
df.AirTime.fillna(df.AirTime.mean(), inplace=True)
df.ActualElapsedTime.fillna(df.ActualElapsedTime.mean(), inplace=True)
df.TaxiOut.fillna(df.TaxiOut.mean(), inplace=True)
df.WheelsOff.fillna(df.WheelsOff.mean(), inplace=True)
df.ArrDelay.fillna(df.ArrDelay.mean(), inplace=True)
df.ArrDel15.fillna(df.ArrDel15.mean(), inplace=True)
df.ArrivalDelayGroups.fillna(df.ArrivalDelayGroups.mean(), inplace=
    True)
df.WheelsOn.fillna(df.WheelsOn.mean(), inplace=True)
```

kiểm tra lại bộ dữ liệu: `df.isnull().sum()`.

```
Airline                                 0
Origin                                  0
Dest                                    0
Cancelled                               0
Diverted                                0
CRSDepTime                              0
DepTime                                 0
DepDelay                                0
ArrTime                                 0
ArrDelayMinutes                         0
AirTime                                 0
CRSElapsedTime                          0
```

```
ActualElapsedTime                          0
Distance                                   0
Year                                       0
Quarter                                    0
Month                                      0
DayofMonth                                 0
DayOfWeek                                  0
Marketing_Airline_Network                  0
Operated_or_Branded_Code_Share_Partners    0
OriginCityName                             0
OriginStateName                            0
DestCityName                               0
DestStateName                              0
DepartureDelayGroups                       0
TaxiOut                                    0
WheelsOff                                  0
WheelsOn                                   0
CRSArrTime                                 0
ArrDelay                                   0
ArrDel15                                   0
ArrivalDelayGroups                         0
ArrTimeBlk                                 0
DistanceGroup                              0
DivAirportLandings                         0
DepTimeBlk_start                           0
DepTimeBlk_end                             0
New_DepTimeBlk                             0
dtype: int64
```

Như vậy các bước tiền xử lý dữ liệu đã xong, việc tiếp theo là đọc hiểu bộ dữ liệu.

5.3 EDA dữ liệu.

Ta phân ra hai nhóm dữ liệu là nhóm chứa hai giá trị và nhóm có nhiều hơn 2 giá trị để có thể đánh giá phân bố dữ liệu. Đối với nhóm chứa hai giá trị, ta phân loại chúng bằng câu lệnh sau:

```
bool_columns = []
for col in df.columns:
    if df[col].nunique()<3:
        bool_columns.append(col)
print(bool_columns)
```

Ta được:

```
['Cancelled',
 'Diverted',
 'DepDelay',
 'Year',
 'DepartureDelayGroups',
 'New_DepTimeBlk']
```

Các biến còn lại sẽ thuộc các biến có 2 giá trị trở lên, ta sẽ tạo một mảng và đưa chúng vào.

```
Num_col = []
for i in df.columns:
    if i not in bool_columns and i not in catego_col:
        Num_col.append(i)
print(Num_col)
```

Ta được:

```
['CRSDepTime',
 'DepTime',
 'ArrTime',
 'ArrDelayMinutes',
 'AirTime',
 'CRSElapsedTime',
 'ActualElapsedTime',
 'Distance',
 'Quarter',
 'Month',
 'DayofMonth',
 'DayOfWeek',
 'TaxiOut',
 'WheelsOff',
 'WheelsOn',
```

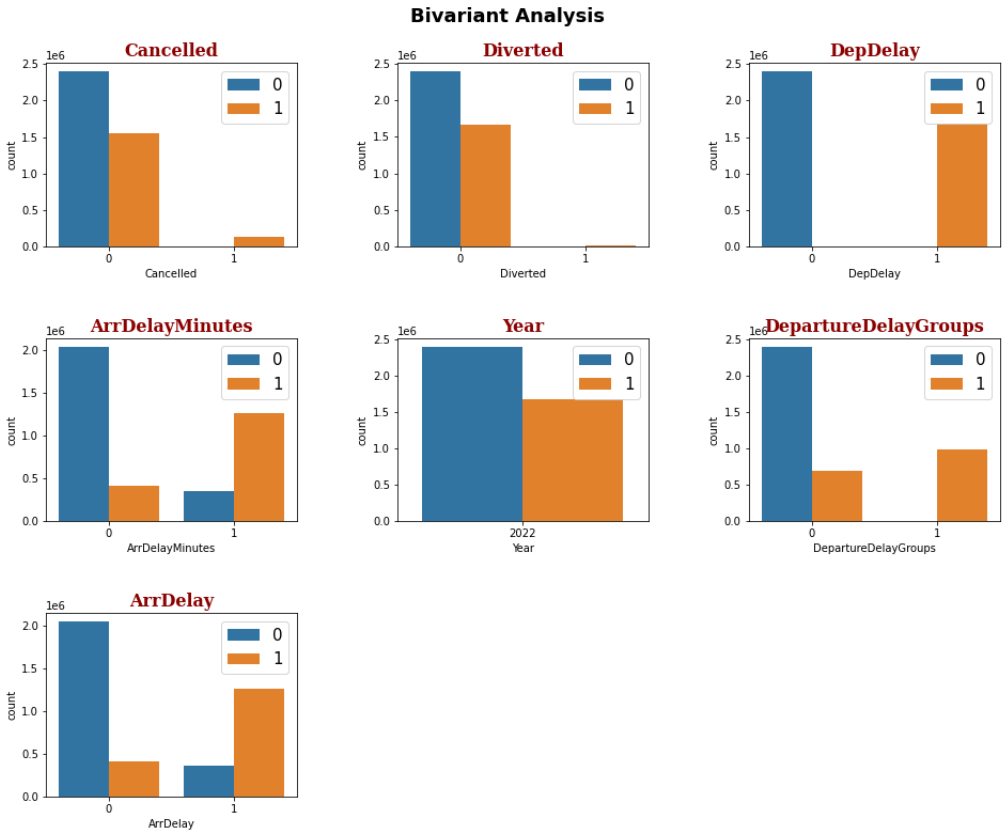
```
 'CRSArrTime',
 'ArrDelay',
 'ArrDel15',
 'ArrivalDelayGroups',
 'DistanceGroup',
 'DivAirportLandings']
```

Để đánh giá trực quan, ta sẽ biểu diễn dữ liệu trên bằng các biểu đồ trực quan để có thể đánh giá phân bổ dữ liệu trong tập dữ liệu ta có.

```
fig = plt.figure(figsize=[16,12])
fig.suptitle('Bivariant Analysis', fontsize=18, fontweight='bold')
fig.subplots_adjust(top=0.92);
fig.subplots_adjust(hspace=0.5, wspace=0.4);
for i ,col in enumerate(bool_columns):
    a = fig.add_subplot(3, 3, i+1)
    a=sns.countplot(x = df[col] , ax=a , hue = df['DepDelay'] )
    a.set_title(col , fontdict={'family': 'serif','color':  'darkred
   ','weight': 'bold','size': 16})
    a.legend(fontsize=15)
```

```
fig = plt.figure(figsize=[16,12])
fig.suptitle('Bivariate Analysis', fontsize=18, fontweight='bold')
fig.subplots_adjust(top=0.92);
fig.subplots_adjust(hspace=0.5, wspace=0.4);
for i ,col in enumerate(Num_col):
    a = fig.add_subplot(5, 5, i+1)
    box_plot=df.boxplot(col)
```

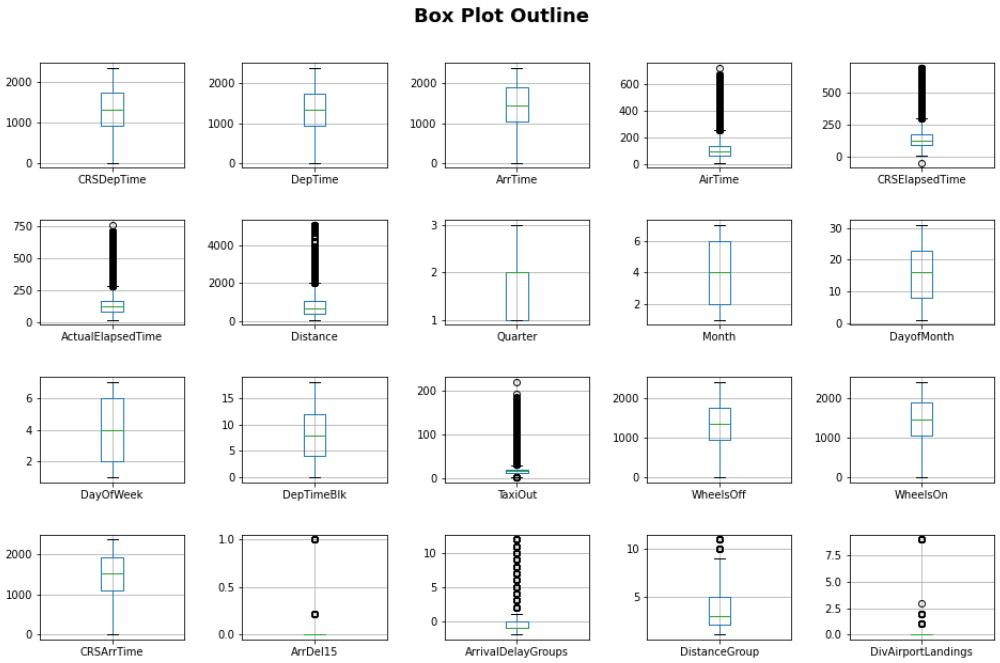
Dự vào hai biểu đồ hình (5.2) và (5.3) ta thấy, đối với nhóm dữ liệu chỉ có hai giá trị 0 và 1 thì mức độ phân xuất hiện của 2 giá trị không chênh lệch quá nhiều. Đối với các biến có từ hai giá trị đổ lên có xuất hiện nhiều giá trị ngoại lai. Để xử lý vấn đề này, em sử dụng khoảng tứ phân vị để loại bỏ các giá trị ngoại lai. Với khoảng tứ phân vị, ta sẽ có khoảng giữa giá trị thứ 25% (Q1) và giá trị thứ 75% (Q3) của dữ liệu. Khoảng tứ phân vị (IQR) sẽ được tính ra như sau: $IQR = Q3 - Q1$. Từ đây ta sẽ xác định các giá trị ngoại lai của biến dữ liệu bằng cách so sánh với các giá trị chăn trên và chặn dưới của khoảng tứ phân vị.



Hình 5.2: Bivariant Analysis

Ngưỡng giá trị chặn trên của IQR: $Q1 - (1.5 * IQR)$, ngưỡng giá trị chặn dưới của IQR: $Q3 + (1.5 * IQR)$. Với các giá trị ngoại lai mà nhỏ hơn ngưỡng chặn dưới thì sẽ được thay thế bằng giá trị chặn dưới và giá trị ngoại lai lớn hơn giá trị chặn trên thi sẽ được thay bằng giá trị chặn trên.

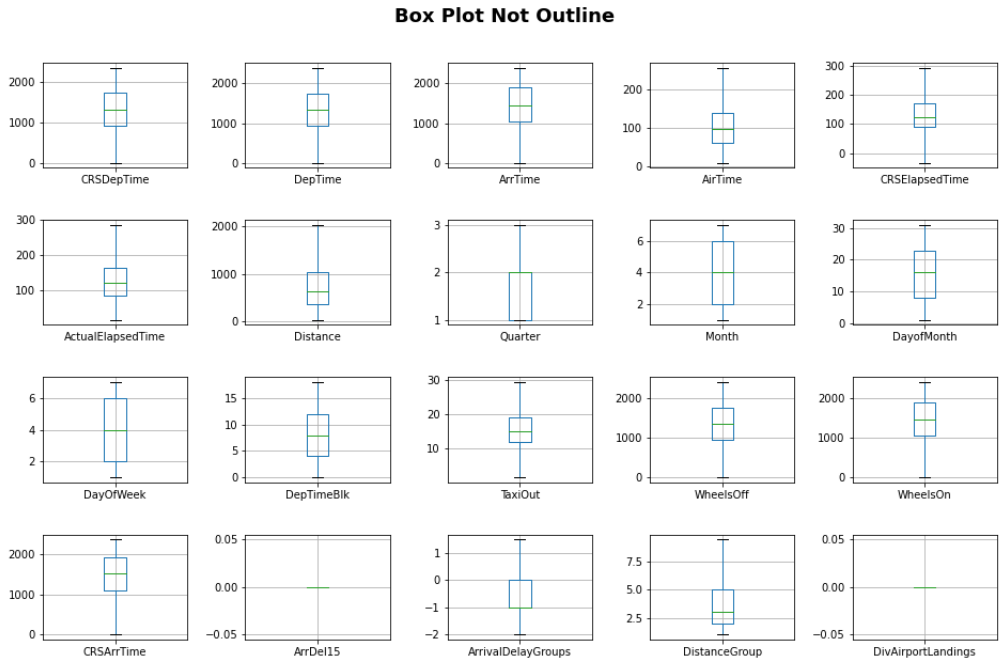
```
dict = {}
for col in Num_col:
    percentile25 = df[col].quantile(0.25)
    percentile75 = df[col].quantile(0.75)
    IQR  = percentile75 - percentile25
    upper_limit = percentile75 + 1.5 * IQR
    lower_limit = percentile25 - 1.5 * IQR
    dict['upper_limit'+ '_' + col] = upper_limit
    dict['lower_limit'+ '_' + col] = lower_limit

for col in Num_col:
```



Hình 5.3: Box Plot Outline

```
    df[col] = np.where(
        df[col] > dict['upper_limit_' + col],
        dict['upper_limit_' + col],
        np.where(
            df[col] < dict['lower_limit_' + col],
            dict['lower_limit_' + col],
            df[col]
        )
    )
```

. Dữ liệu sau khi được xứ lý đã không còn các giá trị ngoại lai nữa và được thể hiện qua biểu đồ (hình 5.4) Tuy vậy, đối với một bộ dữ liệu để training, ngoài các giá trị ngoại lai mà mình cần loại bỏ thì đôi với các giá trị mà có hệ số tương quan thấp hoặc giá trị tương quan không có ý nghĩa với mô hình thì ta cũng có thể loại bỏ để giảm đi chiều của tập dữ liệu traning và có một kết quả chính xác



Hình 5.4: Box Plot Not Outline

hơn. Trong python hỗ trợ cho chúng ta tính hệ số tương quan với biến mục tiêu "DepDelay" như sau:

```
correlation = []
for col in Num_col:
    corr = df[col].corr(df['DepDelay'])
    correlation.append(corr)
    print('He so tuong quan giua '+col+' va DepDelay la: ', corr)
```

kết quả:

```
He so tuong quan giua CRSDepTime va DepDelay la:  0.1869808743283894
He so tuong quan giua DepTime va DepDelay la:  0.2221636603440186
He so tuong quan giua ArrTime va DepDelay la:  0.11011914684991114
He so tuong quan giua AirTime va DepDelay la:  0.07736120146903257
He so tuong quan giua CRSElapsedTime va DepDelay la:
   0.0680012088965457
He so tuong quan giua ActualElapsedTime va DepDelay la:
```

```
    0.07506138429478583
He so tuong quan giua Distance va DepDelay la:  0.07798661629890988
He so tuong quan giua Quarter va DepDelay la:  0.023147649454494084
He so tuong quan giua Month va DepDelay la:  0.029938213241879317
He so tuong quan giua DayofMonth va DepDelay la:
    -0.019558784807029873
He so tuong quan giua DayOfWeek va DepDelay la:  0.052864366297629854
He so tuong quan giua DepTimeBlk va DepDelay la:  0.18787254704342174
He so tuong quan giua TaxiOut va DepDelay la:  0.005421883283318168
He so tuong quan giua WheelsOff va DepDelay la:  0.21399737074812872
He so tuong quan giua WheelsOn va DepDelay la:  0.12207828748607216
He so tuong quan giua CRSArrTime va DepDelay la:  0.14663995650580192
He so tuong quan giua ArrDel15 va DepDelay la:  nan
He so tuong quan giua ArrivalDelayGroups va DepDelay la:
    0.6296689588114268
He so tuong quan giua DistanceGroup va DepDelay la:
    0.0744036337274707
He so tuong quan giua DivAirportLandings va DepDelay la:  nan
```

Ta có thể thấy hai biến 'ArrDel15'và 'DivAirportLandings' không có ý nghĩa trong bộ dữ liệu, cùng với đó 'TaxiOut' có hệ số tương quan quá bé nên ta sẽ loại bỏ ba gía trị này cùng với biến mục tiêu cho tập huấn luyện đầu vào X trong mô hình. Việc xóa bỏ các biến này ta sẽ thực hiện ở bước xây dựng mô hình.

5.4 Xây dựng các mô hình học máy cho dự đoán.

Với mục tiêu ban đầu của mô hình là phân loại chuyến bay bị delay, chính vì vậy ta sẽ lấy biến 'DepDelay' làm biến mục tiêu.

```
X=df.drop(['DepDelay','ArrDel15','DivAirportLandings','TaxiOut'], axis
    =1)
y = df['DepDelay']
```

Ta bắt đầu xây dựng các tập dữ liệu để huấn luyện là test rất đơn giản bằng dòng lệnh sau:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2)}
```

Ở đây, ta chia ra thành hai tập là tập dùng để train (huấn luyện) và tập test (kiểm tra) với tỷ lệ là 80% là tệp train, 20% là tệp test.

5.4.1 Xây dựng mô hình hồi quy Logistic.

Sau khi xây dựng các tập huấn luyện, ta bắt đầu huấn luyện mô hình hồi quy logistic dựa trên tập dữ liệu đã được xử lý.

```
logr = LogisticRegression()
logr.fit(X_train, y_train)
```

Sau khi huấn luyện mô hình với tập dữ liệu huấn luyện, ta bắt đầu chạy mô hình phân loại để dự đoán các chuyến bay delay.

```
Pre = logr.predict(X_test)
```

5.4.2 Xây dựng mô hình cây quyết định với thuật toán CART.

Cùng sử dụng tập huấn luyện ở trên, ta huấn luyện mô hình cây quyết định với thuật toán CART với giới hạn độ sâu như sau:

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion='gini',max_depth=20,random_state
    =0)
dt.fit(X_train, y_train)
```

Trong cây quyết định (Decision Tree), `max_depth` là một tham số quan trọng để giới hạn độ sâu tối đa của cây. Điều này giúp hạn chế độ phức tạp của cây và có thể giúp tránh overfitting. `Max_depth` xác định số lượng tối đa các tầng ẩn của cây quyết định. Mỗi tầng ẩn tương ứng với một cấp độ chia nhánh (branch)

trong cây. Nếu `max_depth` được đặt thành một số nguyên dương, cây quyết định sẽ không được phép phát triển sâu hơn số lượng tầng ẩn này.

Khi giá trị `max_depth` được thiết lập quá cao, cây quyết định có thể trở nên quá phức tạp và overfitting có thể xảy ra, tức là cây học rất chính xác trên dữ liệu huấn luyện nhưng không tổng quát hóa tốt trên dữ liệu mới. Trên thực tế, quá trình tạo ra cây quyết định liên tục chia nhỏ dữ liệu và đưa ra quyết định dựa trên dữ liệu huấn luyện gây ra một sự phù hợp mạnh mẽ với dữ liệu đào tạo.

Khi chạy kiểm thử mô hình, em chọn ra tham số `mmax_depth = 20` như ở trên và cho ra kết quả tối ưu nhất. Ta cần đánh giá xem mô hình có bị Overfiting như đã nêu ở trên hay không bằng cách so sánh hiệu suất huấn luyện của mô hình trên tập dữ liệu huấn luyện và tập dữ liệu dự đoán:

```
#kiem tra overfiting
print('Training set score: {:.4f}'.format(dt.score(X_train, y_train)))
print('Test set score: {:.4f}'.format(dt.score(X_test, y_test)))
```

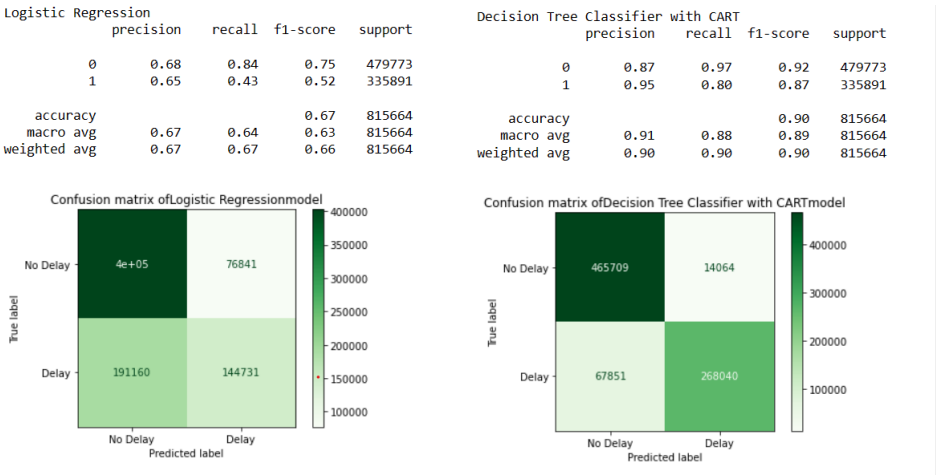
Kết quả

```
Training set score: 0.9336
Test set score: 0.9256
```

Như ta có thể thấy, hiệu quất giữa hai tập huấn luyện và tập dự đoán là khá tương đồng, cho nên ta có thể nhận định rằng với giới hạn độ sâu như trên thì mô hình không bị Overfiting.

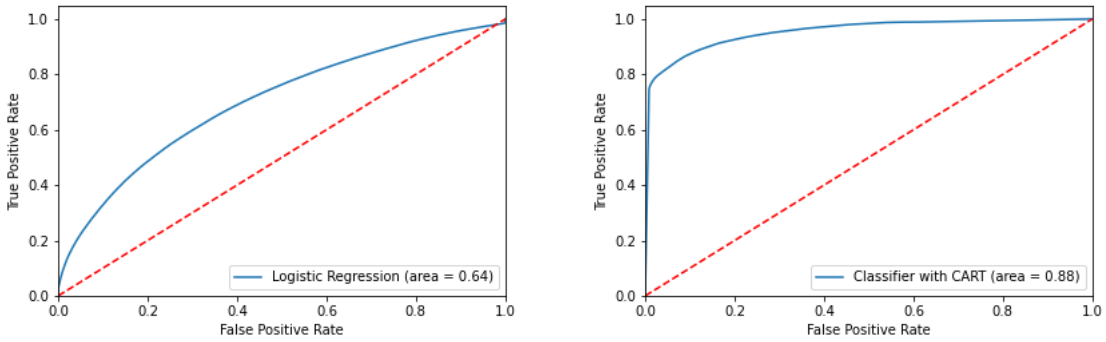
5.5 Đánh giá và so sánh hai mô hình học máy.

Sau đây là kết quả khi chay hai mô hình



Hình 5.5: Kết quả chạy mô hình

Hiệu suất hai mô hình với đường cong ROC.



Hình 5.6: Hiệu suất mô hình

Với kết quả như trên ta có thể nhận xét rằng mô hình học máy cây quyết định hoạt động tốt hơn so với mô hình hồi quy logistic. Dễ thấy, ta so sánh các chỉ số đánh giá mô hình phân lớp cung như biểu đồ về ma trận nhầm lẫn và hiệu suất của hai mô hình thì kết quả đều cho thấy mô hình cây quyết định (CART) áp dụng cho bài toán và bộ dữ liệu này tốt hơn mô hình hồi quy logistic.

Chương 6

Kết luận.

Với mô hình hồi quy logistic và cây quyết định Cart là những mô hình học máy đơn giản và hiệu quả, trong đồ án lần này em mới chỉ áp dụng mô hình với phân loại 2 biến mục tiêu. Trong thực tế, ta có thể gặp phải những bài toán phân loại với nhiều biến mục tiêu hơn. Khi đó bài toán sẽ trở nên phức tạp hơn. Bên cạnh việc phân loại nhiều biến mục tiêu hơn, ta còn có thể xây dựng môt mô hình kết hợp với các mô hình học máy để có thể cho ra một kết quả tốt ưu nhất.

Về phần đồ án lần này, kết quả việc training mô hình tương đối tốt.Tuy nhiên tập dữ liệu thu thập lần này chưa đánh giá được khách quan về nguyên nhân dẫn đến việc máy bay bị delay ví dụ như sự ảnh hưởng của thời tiết, điều này chưa có trong dữ liệu bộ data thu thập. Những thiếu sót trên cần được giải quyết trong tương lai tới.

Ứng dụng các mô hình máy học hiện nay đang được nghiên cứu rất nhiều trên thế giới, trong đó có cả việt nam. Trên đây là bài báo cáo của em về đồ án 2 lần này làm về việc xây dựng hai mô hình máy học hồi quy losgitic và cây quyết định Cart. Ở đây em đã tối ưu hàm mất mát đối với mô hình hồi quy logistic và làm giảm độ phức tạp cho mô hình cây quyết định cart, cùng với đó là đưa ra một vài đánh giá cho hai mô hình máy học. Trên thực tế, ứng với mỗi một bài toán khác nhau thì sẽ có những mô hình phù hợp nhất để giải quyết chứ không

thật sự là thuật toán này tốt hơn thuật toán kia hoặc là kết hợp chúng để đưa ra một kết quả dự đoán tốt .Trong khuôn khổ kiến thức em tìm hiểu được và thời gian nên bài báo cáo còn nhiều hạn chế và thiếu xót. Do vậy rất mong thầy(cô) đóng góp ý kiến để bài báo cáo này được hoàn thiện hơn. Em xin chân thành cảm ơn thầy(cô).

Tài liệu tham khảo

Trong quá trình làm đồ án em đã tham khảo qua những tài liệu: [1–7].

Tài liệu tham khảo

[1] Nguyen-Thi-Thu-Thuy, Suy luan thong ke chuong12-ngttthuy (20201).

[2] Bishop, Pattern Recognition And Machine Learning, 2006.

[3] J. Jiawei Han, Micheline Kamber, Data mining, 2006.

[4] Vu-Huu-Tiep, Logistic regression (2017).
URL https://machinelearningcoban.com/2017/01/27/logisticregression/

[5] Vu-Huu-Tiep, Gradient descent (2017).
URL https://machinelearningcoban.com/2017/01/12/gradientdescent/

[6] (2022). [link].
URL https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=vi

[7] [link].
URL https://www.kaggle.com/datasets/robikscube/flight-delay-dataset-20182022