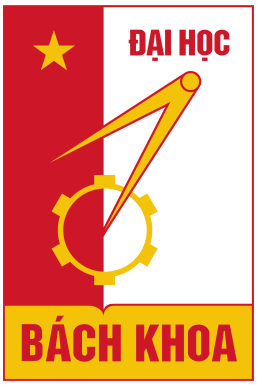
ĐẠI HỌC BÁCH KHOA HÀ NỘI

KHOA TOÁN - TIN



Thuật toán cây quyết định và kỹ thuật Bagging trong bài toán phân lớp

ĐỒ ÁN I

Chuyên ngành: TOÁN TIN

Chuyên sâu: Tin học

Giảng viên hướng dẫn: ThS. Nguyễn Tuấn Dũng

Sinh viên thực hiện: Nguyễn Đình Nam

Mã số sinh viên: 20216859

HÀ NỘI - 2024

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục đích và nội dung của đồ án

Mục đích: Tìm hiểu về kỹ thuật Bagging và thuật toán Random Forest trong Machine Learning.

Nội dung: Trình bày về kỹ thuật Bagging, thuật toán Random Forest và cài đặt chương trình thử nghiệm cho lý thuyết được trình bày trong đồ án.

2. Kết quả đạt được

- Tìm hiểu về bài toán phân loại, phân lớp.
- Các bước tiền xử lý dữ liệu.
- Chuẩn hóa dữ liệu.
- Trực quan hóa dữ liệu.
- Tìm hiểu và ứng dụng các mô hình học máy trong bài toán phân loại: Hồi quy Logistic, Cây quyết định, Random Forest, K-Nearest Neighbors, ...
- Phương pháp Ensemble Learning.
- Các đánh đánh giá mô hình học máy.

3. Ý thức làm việc của sinh viên

Hà Nội, ngày ... tháng ... năm 2024

Giảng viên hướng dẫn

(Ký và ghi rõ họ tên)

ThS. Nguyễn Tuấn Dũng

Mục lục

Mở đầu

Học máy là một lĩnh vực quan trọng có ý nghĩa to lớn trong đời sống của chúng ta tại thời đại 4.0 ngày nay. Học máy được ứng dụng rộng rãi để xử lý các bài toán như thị giác máy tính, phân tích chuỗi thời gian, và nhiều lĩnh vực khác. Trong đó, bài toán phân lớp và phân loại là một trong những ứng dụng quan trọng và phổ biến nhất.

Để nâng cao tính chính xác cho các bài toán phân loại, phương pháp Ensemble Learning là một trong những phương pháp phổ biến được sử dụng. Ensemble Learning giúp kết hợp nhiều mô hình học máy để tạo ra một mô hình mạnh mẽ và chính xác hơn. Một trong các phương pháp của Ensemble Learning là Bagging. Đồ án này sẽ nghiên cứu về Bagging, cũng như một thuật toán sử dụng Bagging khá nổi tiếng đó là Random Forest.

Cấu trúc đồ án:

- Chương 1: Giới thiệu về bài toán phân lớp: Giới thiệu tổng quan về bài toán và các cách đánh giá.
- Chương 2: Thuật toán cây quyết định: Trình bày chi tiết về thuật toán cây quyết định.
- Chương 3: Kỹ thuật Bagging: Trình bày chi tiết về Bagging cũng như Random Forest.
- Chương 4: Chương trình thử nghiệm: Áp dụng các lý thuyết được trình bày ở trên vào một bộ dữ liệu cụ thể.
- Chương 5: Tổng kết và đưa ra hướng phát triển.

Cuối cùng, em xin gửi lời cảm ơn chân thành đến ThS. Nguyễn Tuấn Dũng đã dành thời gian hướng dẫn và tận tình chỉ bảo, đóng góp ý kiến trong quá trình nghiên cứu và thực hiện đồ án. Có thể trong đồ án sẽ có một vài sai sót nho nhỏ nên rất mong

nhận được sự góp ý, chỉ bảo của các thầy, các cô.

Em xin chân thành cảm ơn!.

Hà Nội,. . . ngày . . . tháng năm 2024

Sinh viên thực hiện

Nguyễn Đình Nam

Chương 1

Giới thiệu bài toán phân lớp

1.1 Tổng quan về bài toán phân lớp

1.1.1 Khái niệm

Bài toán phân loại là bài toán thuộc lớp bài toán học có giám sát trong Machine Learning. Bằng việc sử dụng các thuật toán trong Machine Learning để gán nhãn một cách chính xác nhất cho các mẫu từ bộ dữ liệu đầu vào. Một ví dụ đơn giản là dự đoán một người có bị bệnh tiểu đường không thông qua các chỉ số HbA1c, đường huyết,. . . .

1.1.2 Phân loại các bài toán phân lớp

Các bài toán phân lớp được chia thành các loại sau:

- Bài toán phân lớp nhị phân (Binary Classification): Mục đích của bài toán là phân loại dữ liệu thành 2 nhãn lớp. Một số ví dụ ta thường gặp như : phân loại email là thư rác hoặc không? Khách hàng sử dụng dịch vụ sẽ rời đi hay tiếp tục sử dụng?, . . . Thông thường bài toán này dựa trên mô hình dự đoán phân phối xác suất Bernoulli. Một số thuật toán phổ biến được sử dụng gồm: Hồi quy Logistic, Cây quyết định, SVM, . . .
- Phân lớp nhiều lớp (Multi-Class Classification): Nhiệm vụ của bài toán này là phân loại có nhiều hơn 2 nhãn lớp. Từ bộ dữ liệu đầu vào, các mẫu sẽ được phân loại vào một trong các lớp đã biết. Ví dụ: Phân loại khuôn mặt, thực vật, . . . Người ta thường lập mô hình nhiệm vụ phân lớp nhiều lớp với một mô hình dự đoán phân phối xác suất Multinoulli cho mỗi mẫu. Một số thuật toán phổ biến: k-Nearest Neighbors, Cây quyết định, Rừng ngẫu nhiên, Gradient Boosting,. . . Các thuật toán phân lớp nhị phân có thể dùng cho các bài toán phân loại nhiều lớp.

- Phân lớp nhiều nhãn (Multi-label Classification) : Đề cập đến các nhiệm vụ phân lớp có hai hoặc nhiều nhãn lớp, trong đó một hoặc nhiều nhãn lớp có thể được dự đoán cho mỗi mẫu. Bài toán phân loại ảnh là bài toán khá phổ biến trong loại bài toán này. Nhiệm vụ phân loại nhiều nhãn này được dự đoán với một mô hình có nhiều đầu ra và 1 đầu ra tương ứng với một phân phối xác suất Bernoulli. Vậy thì bản chất của nó chính là tạo ta nhiều dự đoán phân loại nhị phân cho mỗi mẫu. Một số thuật toán phổ biến có thể kể đến là: Multi-label Decision Trees, Multi-label Random Forest,. . .
- Phân lớp không cân bằng (Imbalanced Classification): Đề cập đến các nhiệm vụ phân lớp trong đó số lượng ví dụ trong mỗi lớp được phân bố không đồng đều. Một số ví dụ: Xét nghiệm y tế, phát hiện gian lận trong thi cử,. . .

1.2 Cách đánh giá một bài toán phân lớp

1.2.1 Ma trận nhầm lẫn

Ma trận nhầm lẫn (Confusion matrix) của bài toán nhị phân được xác định như sau: Trong đó:

Prediction / Reality	1	0
1	TP	FN
0	FP	TN

Hình 1. Ma trận nhầm lẫn

- TP (True positive): số lượng mẫu được dự đoán là positive và thực tế là positive.
- FP (False positive): số lượng mẫu được dự đoán là positive và thực tế là negative.
- TN (True negative): số lượng mẫu được dự đoán là negative và thực tế là negative.
- FN(False negative): số lượng mẫu được dự đoán là negative và thực tế là positove.

1.2.2 Các chỉ số đánh giá

- Độ chính xác (Accuracy): là đại lượng thể hiện tỷ lệ giữa số lượng mẫu được phân loại đúng so với tổng số mẫu. Tuy nhiên nó không thể phản ánh được đầy đủ hiệu suất của mô hình trong 1 số trường hợp.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

- Độ nhạy (Recall/Sensitivity): là đại lượng đo lường tỷ lệ mẫu dương thực sự đã được phân loại đúng so với tổng số mẫu dương.

$$\text{Recall} = \frac{TP}{TP+FN}$$

- Độ chính xác (Precision): là đại lượng đo tỷ lệ mẫu dương được phân loại đúng so với tổng số mẫu được phân loại là dương.

$$\text{Precision} = \frac{TP}{TP+FP}$$

- Điểm F1 (F1 score): đo lường tỷ lệ mẫu dương thực sự đã được phân loại đúng so với tổng số mẫu dương.

$$F1_{score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- Chỉ số Specificity: là tỷ lệ giữa số lượng mẫu thực sự thuộc lớp âm và được phân loại đúng so với tổng số mẫu thực sự thuộc lớp âm.

$$\text{Specificity} = \frac{TN}{TN+FP}$$

- ROC – AUC:
  - ROC (Receiver Operating Char-acteristic) được sử dụng để đánh giá hiệu quả của mô hình dự đoán. Đường ROC biểu diễn sự tương quan giữa tỷ lệ dương tính và tỷ lệ giả âm tính của mô hình trên các ngưỡng khác nhau.
  - AUC (Area under the curve): là diện tích ở bên dưới của đường ROC.

1.2.3 Độ lệch và phương sai

1. **Độ lệch (Bias)**

- Độ lệch là chỉ số đo lường mức độ sai lệch của mô hình so với thực tế. Ta luôn mong muốn chỉ số này sẽ thấp khi xây dựng mô hình vì giá trị dự đoán sẽ gần với giá trị thực tế. Tuy nhiên trong những bộ dữ liệu lớn, việc mô hình đơn giản quá có thể dẫn tới vấn đề Underfitting. Underfitting là tình trạng khi mô hình học máy không đủ mạnh để hợp lý diễn giải dữ liệu huấn luyện, dẫn đến kết quả là mô hình có độ chính xác thấp trên dữ liệu huấn luyện và có thể không tốt trên dữ liệu kiểm tra. Nguyên nhân là do mô hình của ta quá đơn giản trong khi các dữ liệu có mối quan hệ phức tạp với nhau. Để giảm độ lệch ta có thể sử dụng một số phương án sau : tăng độ phức tạp của mô hình, tìm siêu tham số, đánh giá lại bộ dữ liệu, thu thập thêm dữ liệu và Ensemble Learning.

2. **Phương sai (Variance)**

- Phương sai là chỉ số đo lường mức độ biến động của dự đoán của một mô hình khi được huấn luyện trên các tập dữ liệu khác nhau. Nó đặc trưng cho độ nhạu của mô hình tức là khi ta thay đổi dữ liệu huấn luyện. Một mô hình có phương sai cao có thể dẫn tới việc Overfitting, nơi mô hình "nhớ" dữ liệu huấn luyện 1 cách quá mức và không thể tổng quát hóa cho dữ liệu mới. Vì thế khi gặp những dữ liệu mới, nó sẽ đưa ra một dự đoán thiếu độ chính xác. Để khắc phục điều này ta có thể sử dụng một số cách sau: giảm độ phức tạp của mô hình, sử dụng phương pháp đánh giá chéo, Ensemble Learning.

Chương 2

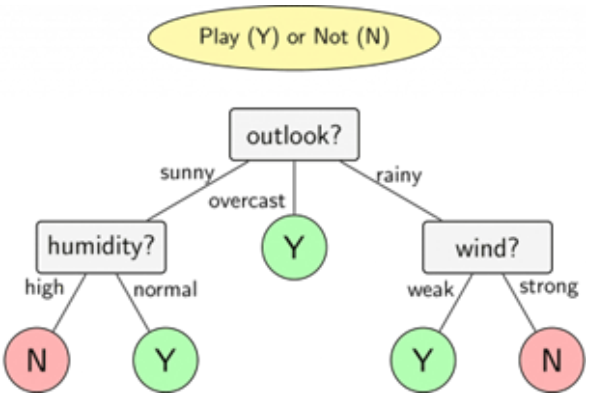
Thuật toán cây quyết định

Cây quyết định (Decision Tree) là một trong những phương pháp vô cùng phổ biến trong Machine Learning, được ứng dụng trong cả bài toán phân loại và hồi quy. Đây là một mô hình mạnh mẽ và sẽ được trình bày ngay dưới đây.

2.1 Định nghĩa

Cây quyết định sử dụng cấu trúc dạng cây để thể hiện một số đường dẫn quyết định có thể có và kết quả của mỗi đường dẫn. Với cách thể hiện này sẽ vô cùng dễ hiểu trong việc sử dụng để mô trả trực quan dữ liệu. Các nút ở dưới cùng gọi là nút lá, mỗi nút lá thể hiện cho các lớp (bài toán phân loại) hoặc giá trị dự đoán (bài toán hồi quy). Các nút trong cây đại diện cho một thuộc tính của bộ dữ liệu và mỗi cạnh từ nút đại diện cho một giá trị trị thuộc tính.

Ví dụ:



Hình 2. Minh họa cây quyết định

2.2 Thuật toán cây quyết định

2.2.1 Thuật toán ID3

Ý tưởng của thuật toán ID3: Ý tưởng của thuật toán là chúng ta sẽ xác định của thuộc tính được xem xét tại mỗi bước. Đối với các bài toán có nhiều thuộc tính và mỗi thuộc tính có nhiều giá trị khác nhau, việc tìm nghiệm tối ưu là khó khăn và ít khả thi. Thay vào đó, có cách khác là tại mỗi bước, thuộc tính tốt nhất được chọn dựa trên 1 tiêu chuẩn nào đó. Ta sẽ chia dữ liệu vào các child node tương ứng với các giá trị của thuộc tính đó rồi áp dụng phương pháp này cho mỗi child node. Cách chọn greedy (tham lam) sẽ giúp ra chọn thuộc tính tốt nhất tại mỗi bước. Tuy nhiên đây có thể không phải cách tối ưu nhưng giúp ta thấy gần với cách làm tối ưu. Sau mỗi câu hỏi, dữ liệu được phân chia vào từng child node tương ứng với các câu trả lời cho câu hỏi đó. Câu hỏi ở đây là một thuộc tính, câu trả lời là giá trị của thuộc tính đó. Nhưng câu hỏi đặt ra ở đây là phân chia như thế nào là tốt, là tối ưu?

Thực tế thì có nhiều hệ số khác nhau mà phương pháp Cây quyết định sử dụng để phân chia nhưng phổ biến nhất là hệ số Entropy.

Entropy

Entropy là một thuật ngữ thuộc lĩnh vực Nhiệt động lực học, là thước đó của sự biến đổi, hỗn loạn hoặc ngẫu nhiên. Sau này, nó được mở rộng sang lĩnh vực nghiên cứu, thống kê. Với n giá trị của biến rời rạc x nhận giá trị lần lượt là $x_1, x_2, ..., x_n$. . Giả sử rằng xác suất để x nhận các giá trị này là $p_i = p(x = x_i)$. Ký hiệu phân phối này là $p = (p_1, p_2, ..., p_n)$. Entropy của phân phối này được định nghĩa như sau:

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

Giả sử tung một đồng xu, entropy được tính như sau:

$$H = -[0.5log(0.5) + 0.5log(0.5)]$$

- P tinh khiết: $p_i = 0$ hoặc $p_i = 1$.Tinh khiết tức là việc lựa chọn biến sao cho sau khi phân chia thì kết quả trả về tại nút con chỉ thuộc về một lớp.

- P vẩn đục: $p_i = 0.5$. Lúc này hàm Entropy đạt đỉnh cao nhất. Vẩn đục là phân phối của các nhãn tại nút con còn khá mập mờ, không có xu hướng thiên về một nhãn nào cụ thể.

Entropy tại một nút lá được tính bằng tổng có trọng số của entropy của từng lớp dữ liệu tại nút đó. Nếu tập dữ liệu của một nút lá chứa N quan sát và có C lớp khác nhau, với số lượng quan sát của lớp thứ i là N_i, entropy tại nút lá đó được tính theo công thức:

$$H(S) = -\sum_{i=1}^{C} \frac{N_i}{N} \log \frac{N_i}{N}$$

Tính hàm số Entropy tại mỗi thuộc tính: với thuộc tính x, các điểm dữ liệu trong S được chia ra nút con $S_1, S_2, ..., S_k$ với số điểm trong mỗi child node lần lượt là $m_1, m_2, ..., m_k$. Ta có:

$$H(x,S) = \sum_{i=1}^{k} \frac{m_i}{N} H(S_i)$$

Định nghĩa infomation gain dựa trên thuộc tính x:

$$G(x,S) = H(S) - H(x,S)$$

Tại mỗi node, thuộc tính đưa chọn xác định dựa trên:

$$x^* = argmax_x G(x,S) = argmin_x H(x,S)$$

Thuộc tính x được lựa chọn sẽ thỏa mãn tổng trọng số Entropy đạt giá trị nhỏ nhất hay nói cách khác khiến cho information gain đạt giá trị lớn nhất.

2.2.2 Ví dụ

outlook	temperature	humidity	wind	play
sunny	hot	high	weak	no
sunny	hot	high	strong	no
overcast	hot	high	weak	yes
rainy	mild	high	weak	yes
rainy	cool	normal	weak	yes
rainy	cool	normal	strong	no
overcast	cool	normal	strong	yes
sunny	mild	high	weak	no
sunny	cool	normal	weak	yes
rainy	mild	normal	weak	yes
sunny	mild	normal	strong	yes
overcast	mild	high	strong	yes
overcast	hot	normal	weak	yes
rainy	mild	high	strong	no

Ví dụ được trích từ cuốn sách Data Mining: Practical Machine Learning Tools and Techniques, trang 11 [3]. Đây có thể coi là một ví dụ khá phổ biến trong các bài giảng về cây quyết định.

Có bốn thuộc tính thời tiết:

- Outlook: có 3 giá trị sunny, overcast, rainy.
- Tempereture: có 3 giá trị hot, cool, mild.
- Humidity: có 2 giá trị high, normal.
- Wind: có 2 giá trị weak, strong.

Entropy tại root node của bài toán:

$$H(S) = \frac{-5}{14}\log(\frac{-5}{14}) - \frac{9}{14}\log(\frac{9}{14}) \approx 0.65$$

Kế tiếp, ta sẽ tính tổng các trọng số của các child node nếu chọn một trong bốn thuộc tính: outlook, temperature, humidity, wind để phân chia dữ liệu.

Giả sử ta chọn outlook. Gọi tập hợp các điểm trong mỗi child node này ứng với 3 giá trị sunny, overcast, rainy lần lượt là S_s, S_0, S_r với m_s, m_o, m_r phần tử.

Ta có:

$$H(S_s) = \frac{-2}{5}\log(\frac{-2}{5}) - \frac{3}{5}\log(\frac{3}{5}) \approx 0.673$$

$$H(S_o) = 0$$

$$H(S_r) = \frac{-2}{5}\log(\frac{-2}{5}) - \frac{3}{5}\log(\frac{3}{5}) \approx 0.673$$

$$H(outlook, S) = \frac{5}{14}H(S_s) + \frac{4}{14}H(S_o) + \frac{5}{14}H(S_r) \approx 0.48$$

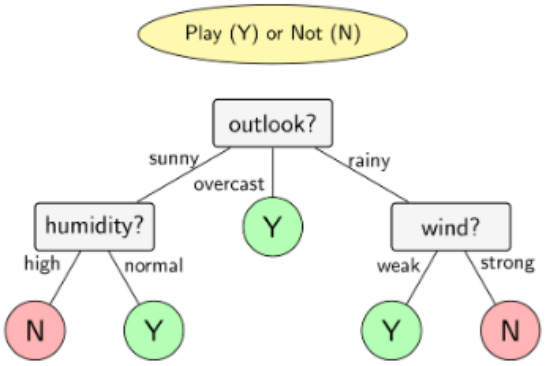
Tương tự như vậy ta sẽ tính được:

$$H(temperature, S) \approx 0.631$$

$$H(humidity, S) \approx 0.547$$

$$H(wind, S) \approx 0.618$$

Như vậy ta sẽ chọn thuộc tính outlook vì H(outlook,S) đạt giá trị nhỏ nhất. Sau đó ta nhận được 3 child node. Tiếp tục áp dụng thuật toán ID3 cho các child node thì ta thu được kết quả sau:



2.2.3 Điều kiện dừng

Trong các thuật toán của Decision Tree nói chung và ID3 nói riêng, nếu tiếp tục phân chia các node chưa tinh khiết, ta thu được moojtj cây mà mọi điểm trong đó đều dự đoán đúng. Khi có cây của ta sẽ phức tạp và sẽ dẫn đến khả năng overfitting. Và để tránh đều đó thì một số cách sau có thể được sử dụng. Tại một node, nếu một trong số các điều kiện dưới đây xảy ra thì không tiếp tục phân chia node đó và coi đó là một leaf node:

- Nếu tại một nút có Entropy bằng 0, nghĩa là mọi điểm trong nút đều thuộc một lớp.
- Nếu tại một nút có số phần tử nhỏ hơn một ngưỡng nào đó. Lúc này, ta chấp nhận có một số điểm phân lớp sai để tránh vấn đề overfitting.
- Nếu khoảng cách từ nút đó đến nút gốc đạt một giá trị nào đó. Việc hạn chế chiều sâu của cây làm giảm độ phức tạp của cây và đồng thời tránh được overfitting.
- Nếu việc phân chia nút đó không làm giảm Entropy quá nhiều.

2.2.4 Ưu nhược điểm của thuật toán cây quyết định

1. Ưu điểm
    - Mô hình dễ hiểu, dễ sử dụng.
    - Dữ liệu đầu vào có thể là là dữ liệu missing, không cần chuẩn hóa hoặc tạo biến giả.

- Có thể làm việc với cả dữ liệu số và dữ liệu phân loại.
- Làm việc với cả bài toán phân loại và hồi quy.
- Cho biết ảnh hưởng của những thuộc tính đến kết quả .

2. Nhược điểm

- Thường xuyên gặp vấn đề overfitting.
- Ít ổn định.
- Phụ thuộc lớn vào dữ liệu đầu vào. Chỉ với một thay đổi nhỏ của dữ liệu, cấu trúc mô hình của cây sẽ thay đổi hoàn toàn.
- Xử lý thiên vị. Thông thường cây quyết định ưu thích những kiểu dữ liệu danh mục.

Chương 3

Kỹ thuật Bagging

3.1 Giới thiệu chung về Ensemble Learning

3.1.1 Ensemble Learning là gì?

Mục tiêu của chúng ta khi sử dụng các mô hình máy học là xây dựng được một mô hình có tính tổng quát cao từ dữ liệu. Tức là nó có thể đưa ra dự đoán một cách chính xác nhất. Phương pháp Ensemble Learning có thể giúp nâng cao tính tổng quát của mô hình bằng cách sử dụng nhiều mô hình máy học "yếu" sau đó sẽ tổng hợp các kết quả từ mô hình yếu để thu được một mô hình "mạnh". Mô hình "yếu" trong trường hợp này được hiểu là những mô hình có độ chính xác không cao khi hoạt động độc lập.

3.1.2 Phân loại Ensemble Learning

Hiện nay phương pháp Ensemble Learning được phân thành 3 loại chính:

- Bagging: là một phương pháp xây dựng một lượng lớn các mô hình (thông thường là cùng loại) dựa trên những mẫu dữ liệu từ tập huấn luyện. Những mô hình này sẽ được huấn luyện độc lập và song song với nhau. Sau đó kết quả cuối cùng sẽ được tổng hợp từ các kết quả của lượng lớn các mô hình nói ở trên.

- Boosting: là một phương pháp xây dựng một lượng lớn các mô hình (thông thường là cùng loại). Mỗi mô hình sẽ học cách sửa những lỗi của mô hình trước (dữ liệu mà mô hình trước dự đoán sai). Sau đó tạo thành một chuỗi các mô hình mà mô hình sau sẽ tốt hơn mô hình trước bởi trọng số được cập nhập qua mỗi mô hình (trọng số của những dữ liệu dự đoán đúng sẽ được giảm xuống, còn những

dữ liệu sai sẽ tăng lên). Kết quả của mô hình cuối cùng sẽ là kết quả được chọn vì kết quả của mô hình sau sẽ tốt hơn kết quả của mô hình trước.

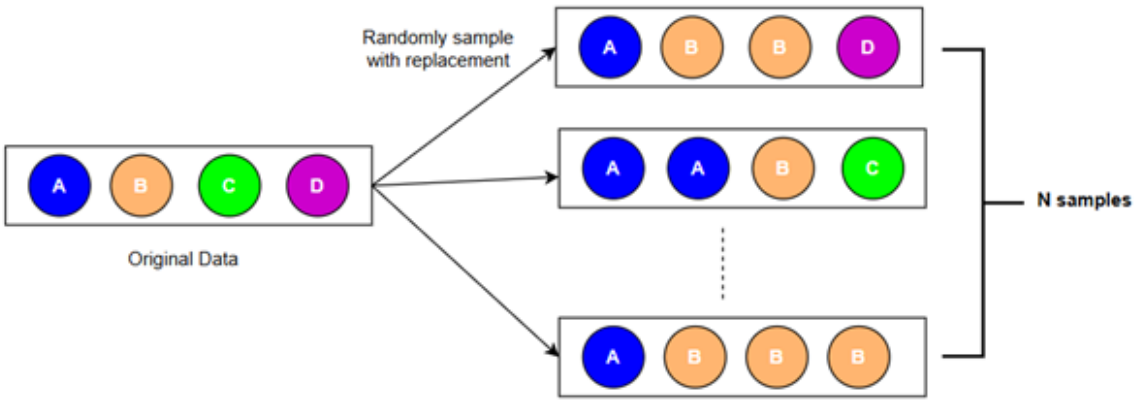
- Stacking: một mô hình Stacking cơ bản thường được phân thành 2 cấp là level-0 modes và level-1 model (hoặc nhiều hơn). Ở level-0 modes (Base-Models), mô hình cơ sở học trực tiếp từ bộ dữ liệu và đưa ra dự đoán cho mô hình cấp sau đó. Ở Level-1 Model (Meta-Model), mô hình học từ các dự đoán của mô hình cơ sở. Sau đó Meta-Model sẽ học cách kết hợp kết quả dự báo của một số mô hình một cách tốt nhất.

Trong 3 biến thể trên thì Bagging giúp ensemble model giảm variance. Còn Boosting và Stacking tập trung vào việc giảm bias.

3.2 Kỹ thuật Bagging

3.2.1 Boostrapping

Bostrapping là một phương pháp lấy mẫu trong học máy. Mẫu mới sẽ được tạo ra bằng cách từ tập dữ liệu ban đầu, một phần từ trong đó sẽ được lặp lại và được chọn ngẫu nhiên vào mẫu mới. Một phần từ có thể được chọn nhiều lần hoặc một số phần từ không được chọn vào mẫu mới. Cứ như vậy cho đến khi thu được số lượng mẫu mới bằng với số lượng mẫu của tập dữ liệu gốc.



Hình 4. Bostrapping.

3.2.2 Aggregating

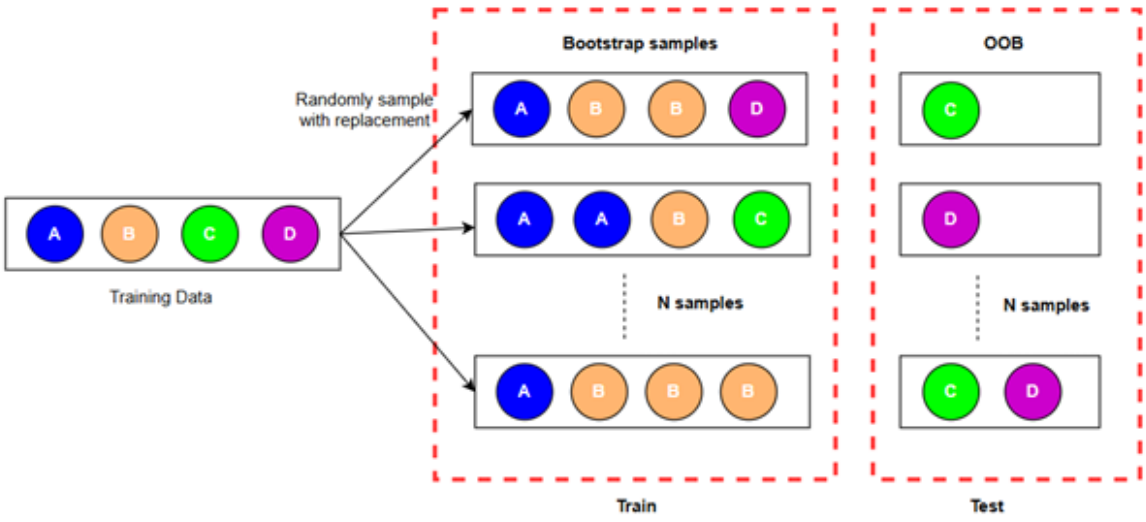
Aggregating là quá trình trong Bagging để tổng hợp các kết quả từ các mô hình "yếu" để cho ra kết quả của mô hình "mạnh".

Các phương pháp tổng hợp:

- Majority voting: đối với mẫu dữ liệu sẽ được phân loại bởi nhiều mô hình riêng lẻ và các dự đoán từ các mô hình này được tổng hợp lại và dự đoán cuối cùng được xác định bằng cách chọn dự đoán mà được đa số các mô hình đồng thuận.

- Average: đây là một phương pháp tính trung các kết quả từ kết quả của các mô hình "yếu". Với mỗi mô hình "yếu", kết quả dự đoán cho mỗi mục tiêu sẽ được tính trung bình. Kết quả cuối cùng thu được sẽ là giá trị trung bình của các kết quả dự đoán từ tất cả các mô hình con.

3.2.3 Out of bag (OOB)



Hình 5. Out of bag.

Khi ta thực hiện phương pháp lấy mẫu Bootstrapping thì có một số điểm dữ liệu sẽ không được chọn vào mẫu để huấn luyện mô hình, những điểm dữ liệu này sẽ được chọn để kiểm tra mô hình và chúng được gọi là những dữ liệu Out of bag (OOB).

Vì những dữ liệu OOB này hoàn toàn mới lạ với dữ liệu huấn luyện nên kết quả

mô hình sẽ đáng tin hơn khi gặp những mẫu này.

3.2.4 Thuật toán Bagging

Bagging là phương pháp kết hợp nhiều mô hình học máy có độ chính xác cao nhưng đổi lại có phương sai lớn để tăng tính ổn định. Điều này giúp tránh khỏi hiện tượng overfitting.

Ý tưởng của thuật toán này có thể hiểu thông qua 3 bước cơ bản sau: Input: Cho bộ dữ liệu huấn luyện $D = (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$. Output: Đối với bài toán nhị phân ta muốn quan sát phân lớp của x

- Bước 1: Mỗi mẫu gồm B tập dữ liệu, mỗi tập gồn N quan sát từ tập gốc D bằng phương pháp lấy mẫu Boostrapping.
- Bước 2: Huấn luyện B bộ dữ liệu thu được từ bước 1.
- Bước 3: Từ kết quả huấn luyện của bước 2. Tiến hành tổng hợp các kết quả bằng phương pháp Aggregating.

Algorithm 1: Thuật toán Bagging [1]

Input: Tập dữ liệu huấn luyện (x_i, y_i), trong đó $x_i \in \mathbb{R}^n$ và $y_i \in \mathbb{R}$
K là số lượng mô hình yếu.
Chọn L là mô hình huấn luyện.
Output: Kết quả dự đoán $\hat{y}_{test}$

$E \leftarrow \{\}$; `// Tập lưu kết quả đã được huấn luyện`
for *mỗi mô hình yếu* $i = 1$ *to* K **do**
    Lấy mẫu theo phương pháp Bootstrap từ tập huấn luyện ban đầu T ta thu được tập huấn luyện Bootstrap T_B ;
    $h_{ti} \leftarrow L(T_{B_i})$; `// Học giả thuyết của mô hình áp dụng` L `trên mẫu Bootstrap`
    E $(E = E \cup h_{ti})$;
Với mỗi mẫu kiểm tra chưa thấy trước x;
Đánh giá $E = \{h_1, h_2, \ldots, h_K\}$ trên x;
Với mỗi mô hình, tính trọng số thứ k: $w_i = \frac{\hat{r}_i}{\sum_{k=1}^{K} \hat{r}_i}$;
Dự đoán mục tiêu thu được: $\hat{y}_{test} = \frac{1}{K} \sum_{i=1}^{K} (\hat{y}_i + w_i \hat{r}_i)$;
return $\hat{y}_{test}$

3.3 Thuật toán Random Forest

3.3.1 Khái niệm

Random Forests là thuật toán học có giám sát (supervised learning). Nó có thể được sử dụng cho cả phân lớp và hồi quy. Đây là một phương pháp sử dụng đồng thời

Bagging và Decision Tree. Ban đầu nó sẽ tạo ra nhiều cây khác nhau để huấn luyện trên một bộ dữ liệu. Sau đó ta sẽ sử dụng Bagging để tạo ra một tập các cây và tổng hợp kết quả từ các cây này để thu được kết quả cuối cùng.

3.3.2 Thuật toán

Thuật toán Random Forest [2]:

Input: Dữ liệu huấn luyện và dữ liệu kiểm tra

Output: Dữ liệu được phân loại

- Bước 1: Lấy mẫu Bootstrap trích xuất n tập huấn luyện con có cùng dung lượng với tập dữ liệu huấn luyện. Mỗi lần dữ liệu không được lấy sẽ ghi nhận ở túi OOB.

- Bước 2: n tập con được tạo thành n cây quyết định và rừng ngẫn nhiên ban đầu được hình thành. Tính toán và tìm ra cây phân chia tốt nhất.

- Bước 3: Tính toán mối tương quan giữa hai cây quyết định trong rừng ngẫu nhiên ban đầu. Hai cây quyết định sẽ được giữ lại nếu mối tương quan dưới một ngưỡng nào đó.Nếu mối tương quan trên ngưỡng đó, cắt bỏ cây quyết định phân loại kém hơn và giữ lại cây tốt hơn cho lần lặp tiếp theo. Sau khi tính toán, các cây quyết định còn lại sẽ tạo thành một rừng ngẫu nhiên mới.

- Bước 4: : Trọng số của mỗi cây quyết định trong rừng ngẫu nhiên mới được tính toán theo độ chính xác của dữ liệu ngoài túi (OOB). Nhập tập dữ liệu kiểm tra và bỏ phiếu kết quả phân loại của cây quyết định theo trọng số để thu được kết quả phân loại.

3.3.3 Các tham số của Random Forest

n_estimators

- Là tham số thể hiện số lượng cây quyết định trong Randon Forest. Nếu giá trị này quá nhỏ, mô hình sẽ quá đơn giản và dẫn đến underfitting. Ngược lại nếu

quá lớn, mô hình qusa phức tạp, thời gian chạy lâu và dẫn đến overfitting. Thông thường giá trị sẽ nằm trong (100, 1000) và default = 100.

max_features

- Là tham số chỉ mỗi cây chỉ chọn một tập nhỏ các thuộc tính trong quá trình xây dựng.
- Bộ dữ liệu ban đầu có n thuộc tính thì:
- Với bài toán hồi quy: $max\_features = \frac{n}{3}$
- Với bài toán phân lớp: $max\_features = \sqrt{n}$

max_depth

- Là độ sâu tối đa của cây quyết định trong mô hình. Cây sẽ không phát triển nữa khi đạt độ sâu này.

min_samples_split

- Là số lượng mẫu tối thiểu cần thiết để phân chia một nút nội bộ của cây . Nếu số lượng quan sát trong một nút là nhỏ hơn hoặc bằng min_samples_split, nút đó sẽ không được phân tách và trở thành một nút lá.
- Giảm giá trị này có thể dẫn đến cây có kích thước lớn hơn, giúp tránh overfitting.

min_samples_leaf : Là số lượng mẫu tối thiểu cần thiết trong mỗi lá của cây.

random_state

- Được sử dụng để đảm bảo kết quả có thể tái tạo.
- Nếu đặt là một giá trị cố định, kết quả sẽ giống nhau mỗi khi mô hình được huấn luyện.

Còn nhiều các tham số khác nữa nhưng trên dây là các tham số thường được sử dụng.

3.4 Tối ưu tham số của Random Forest

3.4.1 Grid Search

Đây là quy trình tự động tìm kiếm qua một tập hợp các giá trị tham số được xác định trước để xác định cấu hình tốt nhất cho mô hình dựa trên các kết quả hiệu suất được đo lường trên tập dữ liệu kiểm tra.

Cụ thể, trong Grid Search, chúng ta xác định một lưới các giá trị tham số mà chúng ta muốn thử nghiệm. Ví dụ, cho thuật toán Random Forest, chúng ta có thể xác định một lưới các giá trị cho các tham số như n_estimators, max_depth, min_samples_split, và các tham số khác. Sau đó, grid search sẽ tạo ra tất cả các tổ hợp có thể của các giá trị tham số và huấn luyện mô hình với các tổ hợp này.

Cuối cùng, Grid Search sẽ chọn ra tổ hợp tham số mà cho kết quả tốt nhất trên tập dữ liệu kiểm tra.

3.4.2 Random Search

Khác với Grid Search, trong Random Search, chúng ta không xác định một lưới các giá trị tham số cụ thể để thử nghiệm. Thay vào đó, chúng ta chọn ngẫu nhiên các giá trị từ một phân phối xác định trước cho mỗi tham số và đánh giá hiệu suất của mô hình với các tổ hợp tham số này. Random Search có thể là một phương pháp hiệu quả trong việc tìm ra các giá trị tham số tốt nhất với một số lượng lớn các tham số hoặc khi không có thông tin rõ ràng về đâu là các giá trị tham số tốt nhất. Đồng thời, Random Search cũng giúp kiểm tra hiệu suất của mô hình trên một phạm vi rộng hơn của không gian tham số so với Grid Search.

Tuy nhiên điều này sẽ dẫn đến việc thời gian tìm kiếm tham số tốt nhất sẽ lâu hơn khá nhiều so với Grid Search.

Chương 4

Chương trình thử nghiệm

Tại phần này, ta sẽ xây dựng chương trình thử nghiệm với bộ dữ liệu Customer Telecom Churn áp dụng mô hình phân loại Decision Tree, kỹ thuật Bagging và Random Forest đồng thời so sánh kết quả với một số mô hình: hồi quy Logistic, K-Nearest Neighbors.

Chi tiết về kết quả và quá trình được trình bày ở: https://colab.research.google.com/drive/13ZZ33BqK8T7CxcBonf-MO4MHnIOWLyM4?authuser=2

4.1 Giới thiệu về bộ dữ liệu

Bộ dữ liệu Customer Telecom Churn gồm có 3333 dòng và 20 cột:

State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	Total night charge	Total intl minutes	Total intl calls	Total intl charge	Customer service calls	Churn
KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10.0	3	2.70	1	False
OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3	3.70	1	False
NJ	137	415	No	No	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32	12.2	5	3.29	0	False
OH	84	408	Yes	No	0	299.4	71	50.90	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	2	False
OK	75	415	Yes	No	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	3	False

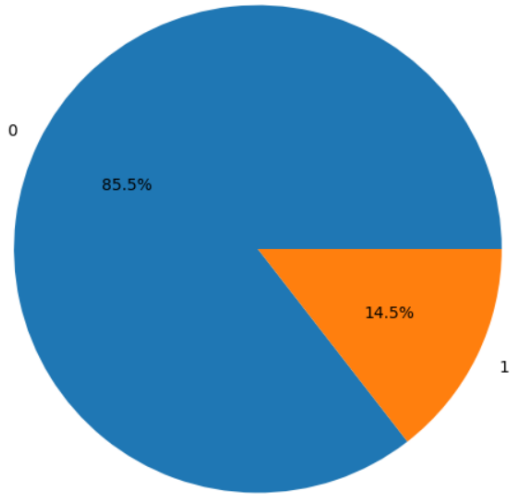
Hình 6. Thông tin về bộ dữ liệu

Sau đó ta tiến hành một số thao tác tiền xử lý dữ liệu cơ bản như: chuyển đổi dạng dữ liệu, xóa trùng lặp, xử lý dữ liệu trống, xử lý ngoại lai,... và ta thu được các kết quả dưới đây:

```
State                     int32
Account length            int64
Area code                 int64
International plan        int64
Voice mail plan           int64
Number vmail messages     int64
Total day minutes       float64
Total day calls           int64
Total day charge        float64
Total eve minutes       float64
Total eve calls           int64
Total eve charge        float64
Total night minutes     float64
Total night calls         int64
Total night charge      float64
Total intl minutes      float64
Total intl calls          int64
Total intl charge       float64
Customer service calls    int64
Churn                     int64
dtype: object
```

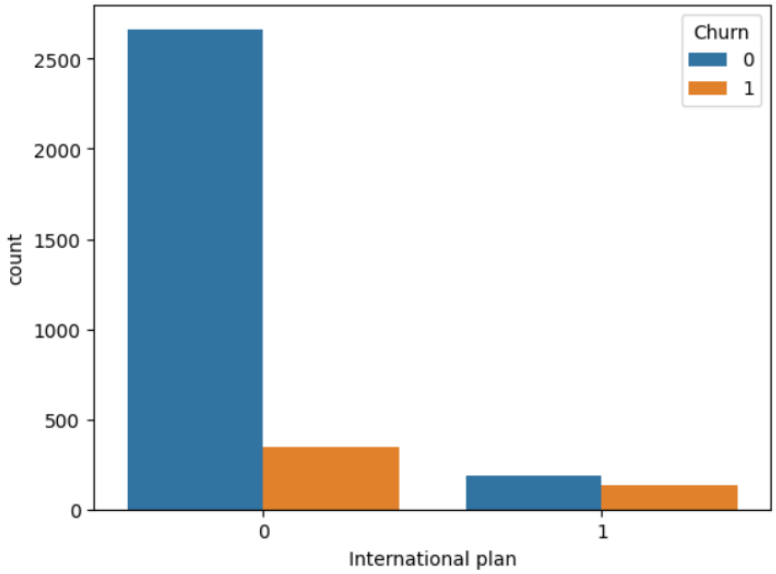
Hình 7. Thông tin về các trường dữ liệu sau khi xử lý

Tiến hành khám phá dữ liệu ta thu được một số kết quả sau:

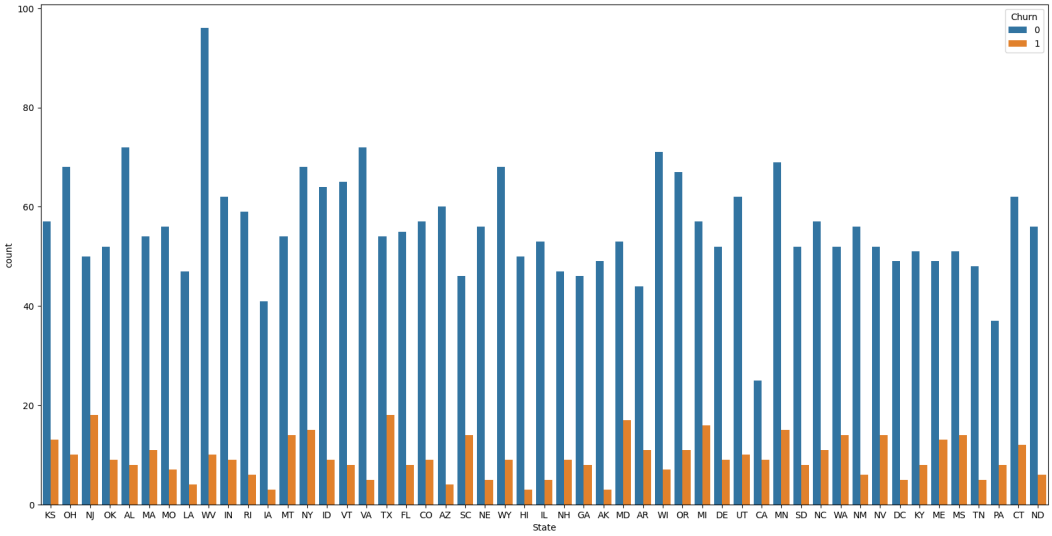


Hình 8. Biểu đồ tỷ trọng khách hàng rời mạng

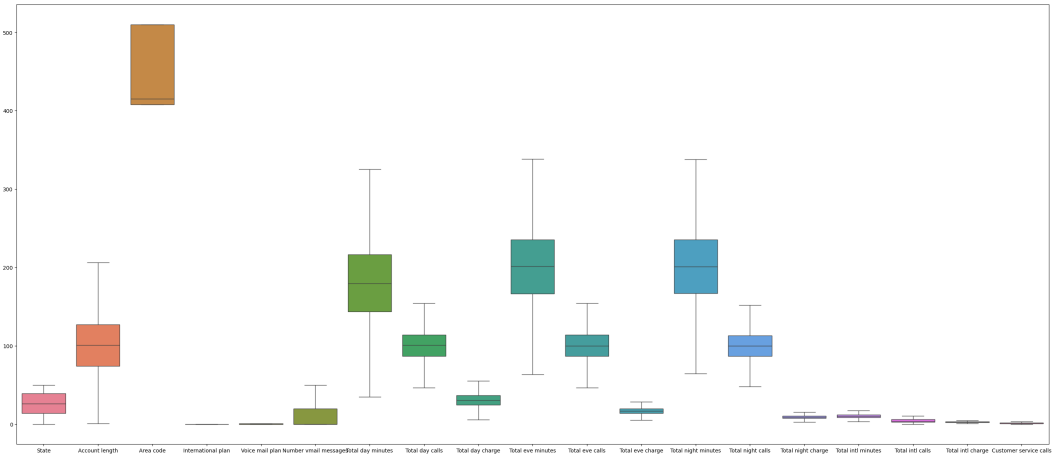
Theo quan sát, có khoảng $85,5\%$ khách hàng không rời mạng và có khoảng $14,5\%$ khách hàng rời đi.



Hình 9. Biểu đồ số lượng khách hàng sử dụng dịch vụ quốc tế



Hình 10. Biểu đồ số lượng khách hàng theo từng vùng



Hình 11.Kết quả sau khi xử lý ngoại lai

4.2 Kết quả chạy chương trình

Dưới đây là các mô hình sẽ được sử dụng trong phần báo cáo này, chúng ta sẽ cung cấp cho nó 1 lưới các tham số và sẽ sử dụng GridSearch và RandomSearch để tìm ra siêu tham số cho các mô hình cũng như kết quả với những siêu tham số đó.

- **Decision Tree**
  - $max\_depth : [1, 3, 5, 10, 20]$
  - Chỉ số xác định độ sâu tối đa của cây.
- **Random Forest**
  - $n\_estimators : randint(100, 1000)$
  - Số lượng cây sử dụng.
  - $max\_depth : range(3, 10)$
  - Độ sâu tối đa của mỗi cây trong Random Forest.
  - $max\_features : [round(math.sqrt(num_columns))]$
  - Đây là số lượng đặc trưng được chọn ngẫu nhiên cho mỗi lần split trong cây quyết định và bằng căn bậc 2 số đặc trưng.
- **KNeighbors**
  - $n\_neighbors : range(1, 20)$
  - Là số lượng hàng xóm gần nhất để sử dụng

- $weights : ['uniform', 'distance']$

· Xác định cách tính trọng số của các hàng xóm gần nhất.

◇ 'uniform': mỗi hàng xóm có cùng một trọng số.

◇ 'distance': các hàng xóm gần hơn sẽ có trọng số lớn hơn, giảm dần theo khoảng cách.

- $metric : ['euclidean', 'manhattan', 'minkowski']$

· Xác định phương pháp đo lường khoảng cách giữa các điểm dữ liệu. Gồm: khoảng cách Euclid, khoảng cách Manhattan, khoảng cách Minkowski.

- **LogisticRegression**

- $solver : ['liblinear', 'lbfgs', 'newton-cg']$

· Xác định thuật toán tối ưu hóa được sử dụng để tìm giá trị tối ưu của hàm mất mát trong quá trình huấn luyện.

- $C : [0.01, 0.1, 1, 10, 100]$

· Xác định sức mạnh của đạo hàm điều chuẩn trong quá trình huấn luyện.

Kết quả sau khi thực thi mô hình cho ở dưới đây:

Bảng 4.1: So sánh các thuật toán phân loại

Thuật toán	Precision	Recall	F1_Score	Accuracy
Decision Tree	0.9	0.73	0.78	0.91
Random Forest	0.93	0.75	0.81	0.92
KNeighbors	0.86	0.62	0.66	0.88
LogisticRegression	0.85	0.55	0.56	0.87

Ta quan sát Bảng 4.1 thấy rằng Random Forest cho ra kết quả tốt nhất trong các thuật toán phân loại kể trên và thấp nhất là Logistic Regression. Điều này là có thể dễ dàng lý giải được vì thuật toán Random Forest sử dụng một lượng lớn cây quyết định cũng như sử dụng lấy mẫu Bootstrap để huấn luyện cho từng cây cho nên khiến cho thuật toán này trở nên vô cùng phức tạp và tồn thời gian chạy chương trình. Bảng 4.2 đây là thời gian thực thi cho đến khi in ra kết quả của mỗi thuật toán:

Bảng 4.2: Thời gian thực thi các thuật toán

Thuật toán	Thời gian huấn luyện(s)	Thời gian dự đoán(s)
Decision Tree	0.8010	0.0000
Random Forest	87.4652	0.0398
KNeighbors	8.4066	0.0873
LogisticRegression	0.7417	0.0000

Với kết quả ở trên thì thời gian thực thi thuật toán RandonForest là mất nhiều thời gian nhất. Thuật toán này có kết quả thực thi lâu hơn rất nhiều so với 3 mô hình còn lại.

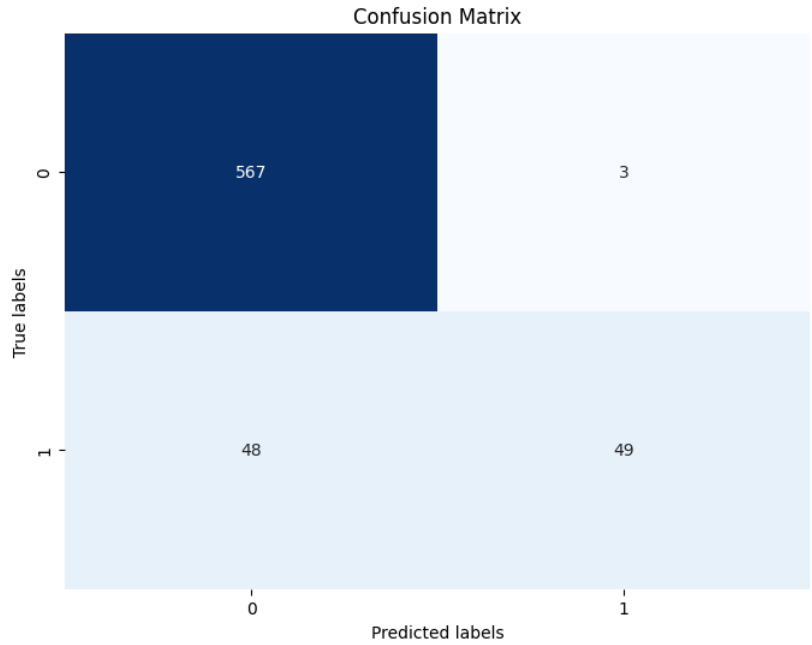
Tiếp theo ta sẽ so sánh độ lệch và phương sai của các thuật toán. Bảng 4.3 cho ta kết quả:

Bảng 4.3: Độ lệch và phương sai

Thuật toán	Độ lệch	Phương sai
Decision Tree	0.0858935711	0.0000647829
Random Forest	0.0840230200	0.0000952134
KNeighbors	0.1230284377	0.0000269878
LogisticRegression	0.1369093043	0.0000042346

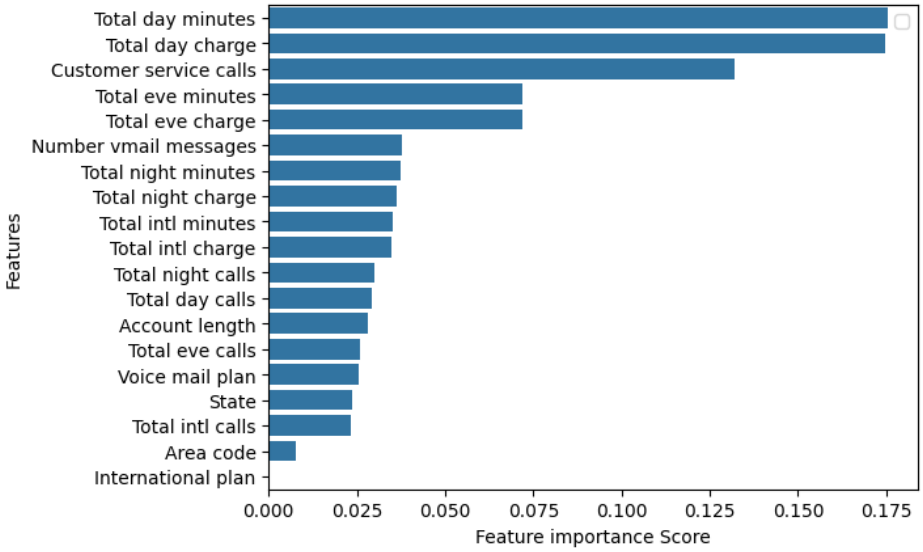
Quan sát Bảng 4.3 ta thấy rằng thuật toán Decision Tree, Bagging và RandomForest cho ta độ lệch thấp hơn so với 2 thuật toán còn lại. RandomForest cho phương sai cao nhất vì đây là mô hình tốt nhất, cho kết quả sát với thực tế nhất trong các mô hình trên.

Ma trận nhầm lẫn của thuật toán RandomForest:



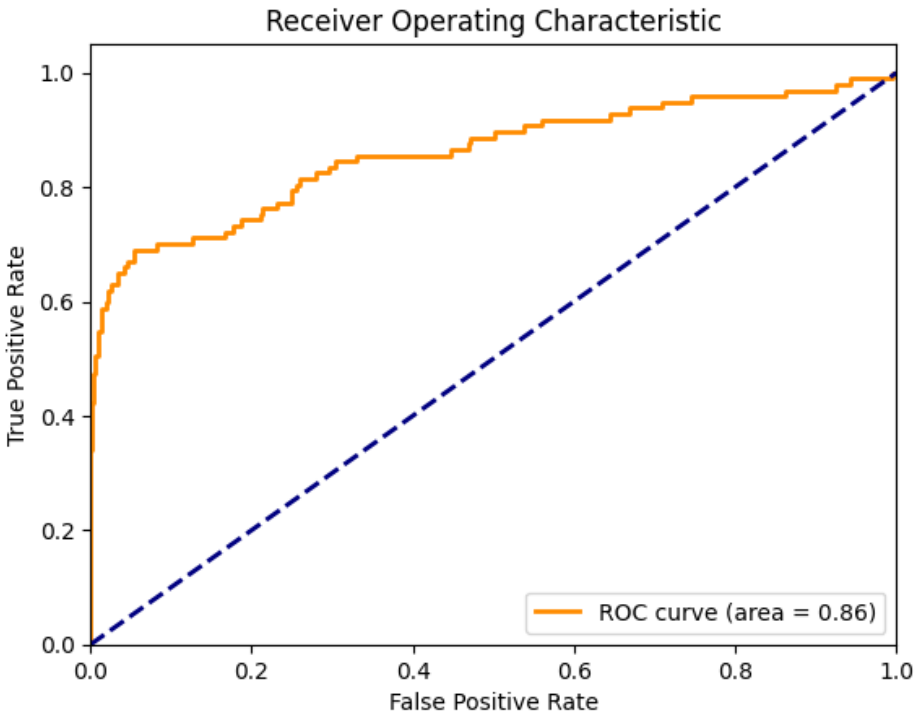
Hình 10. Ma trận nhầm lẫn

Độ quan trọng của các thuộc tính:



Hình 11.Độ quan trọng của các thuộc tính

Quan sát Hình 11 ta thấy rằng thuộc tính 'Total day minutes', 'Total day charge', 'Customer service calls' là 3 thuộc tính có độ quan trọng cao nhất vượt trội hơn so với các thuộc tính còn lại, tuy nhiên thuộc tính 'Area code' có độ quan trọng thấp nhất so với toàn bộ các thuộc tính.



Hình 12.Đường ROC và AUC

Tổng kết lại ta thấy rằng với bộ dữ liệu Customer Telecom Churn khi sử dụng thuật toán RandomForest để tiến hành phân loại cho ra các kết quả đã trình bày ở trên thì đây là một mô hình phân loại tốt.

Chương 5

Tổng kết

Mục tiêu đồ án: Tìm hiểu về kỹ thuật Bagging và thuật toán Random Forest trong Machine Learning.

Kết quả đạt được

1. Giới thiệu tổng quan và phân loại bài toán phân loại
2. Thuật toán cây quyết định
3. Giới thiệu về Emsemble Learning, tìm hiểu và trình chi tiết về kỹ thuật Bagging và thuật toán Random Forest.
4. Tiến hành cài đặt chương trình thử nghiệm 3 mô hình Decision Tree, Bagging, Random Forest và tiến hành so sánh với 2 mô hình khác KNeighbors, Logistic Regression dựa trên một số tiêu chí so sánh.

Kỹ năng đạt được

1. Nâng cao khả năng tự học, đọc tài liệu khoa học, báo cáo,...
2. Bổ sung các kỹ năng, kiến thức liên quan đến chuyên ngành và công việc tương lai
3. Nâng cao khả năng trình bày, gõ báo cáo,....
4. Tư duy giải quyết bài toán,....

Tài liệu tham khảo

[1] Ghoggali, N., Douak, F., & Ghoggali, W. (2022). Towards a NIR Spectroscopy ensemble learning technique competing with the standard ASTM-CFR: An optimal boosting and bagging extreme learning machine algorithms for gasoline octane number prediction. Optik, 257, 168813.
https://doi.org/10.1016/j.ijleo.2022.168813

[2] Vincent, S. S. M., & Duraipandian, N. (2024). Detection and prevention of sinkhole attacks in MANETS based routing protocol using hybrid AdaBoost-Random forest algorithm. Expert Systems With Applications, 249, 123765.
https://doi.org/10.1016/j.eswa.2024.123765

[3] Nordhausen, K. (2014). An Introduction to Statistical Learning—with Applications in R by Gareth James, Daniela Witten, Trevor Hastie & Robert Tibshirani. International Statistical Review, 82(1), 156–157.
https://doi.org/10.1111/insr.12051_19

[4] Grus, J. (2015). Data Science from Scratch: First Principles with Python.
https://www.amazon.com/Data-Science-Scratch-Principles-Python/dp/1492041130