TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Đồ ÁN I

Ứng dụng kỹ thuật Bagging trong bài toán phân lớp

HOÀNG TRUNG CHIếN

chien.ht194922@sis.hust.edu.vn

Chuyên ngành: Toán tin

Giảng viên hướng dẫn: ThS. Nguyễn Tuấn Dũng

Bộ môn: Toán tin

Viện: Viện Toán Ứng dụng và Tin học

Chữ ký của GVHD

HÀ NộI, 2022

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

—————o0o—————

Ứng dụng kỹ thuật Bagging trong bài toán phân lớp

ĐỒ ÁN I

Chuyên ngành: Toán tin

Giảng viên hướng dẫn: ThS. Nguyễn Tuấn Dũng
Sinh viên thực hiện: Hoàng Trung Chiến
Mã sinh viên: 20194922
Lớp: CTTN Toán-Tin K64

HÀ NỘI, 2022

Mục lục

Danh sách hình vẽ

Mở đầu

Học máy đang dần dần len lỏi vào mọi lĩnh vực của đời sống, đặc biệt là trong thời đại công nghiệp 4.0. Học máy là một phần không thể thiếu trong các bài toán phân tích dữ liệu nhằm khai thác được những thông tin trong quá khứ và tiên lượng được những hành vi hay kết quả sẽ xảy ra trong tương lai. Trong thời gian gần đây, người ta đang tìm cách nâng cao sự hiệu quả của các mô hình học máy, một trong những cách đã được tìm ra và sử dụng hiệu quả đó là Ensemble Learning, hay hiểu đơn giản là việc kết hợp các mô hình học máy lại với nhau. Một trong những phương pháp tiêu biểu nhất của Ensemble Learning đó chình là Bagging.

Đồ án này nghiên cứu tìm hiểu về bài toán phâp lớp cùng với một phương pháp Ensemble Learning đó là Bagging, để xem cách thức hoạt động và lợi ích của phương pháp này. Một trong các thuật toán của phương pháp Bagging đó là Random Forest cũng sẽ được xem xét trong đồ án này.

Đồ án triển khai vấn đề trong các chương sau đây:

- **Chương 1. Bài toán phân lớp:** Chương đầu sẽ trình bày trình bày tổng quan về bài toán phân lớp cùng các cách để đánh giá một bài toán phân lớp.
- **Chương 2. Phương pháp Bagging:** Chương 2 sẽ trình bày các chi tiết về cách xây dựng và đánh giá một mô hình sử dụng Bagging.
- **Chương 3. Chương trình minh hoạ:** Áp dụng phương pháp cho một bộ dữ liệu cụ thể, qua kết quả của chương trình để chỉ ra ưu điểm của phương pháp Bagging.

- **Chương 4. Kết luận:** Chương này được dành cho các đánh giá và kết luận về kết quả đã đạt được trong đồ án, cũng như các hướng phát triển tiếp theo.

Để hoàn thành được đồ án lần này, em xin gửi lời cảm ơn chân thành đến ThS. Nguyễn Tuấn Dũng đã hướng dẫn và chỉ bảo em tận tình trong quá trình nghiên cứu và thực hiện đồ án. Trong quá trình nghiên cứu và soạn thảo, vì thời gian và điều kiện có hạn nên đồ án không thể tránh khỏi những sai sót nhất định. Rất mong nhận được sự đóng góp, chỉ bảo và ý kiến của các quý thầy cô.

Trân trọng cảm ơn.

Chương 1

Bài toán phân lớp

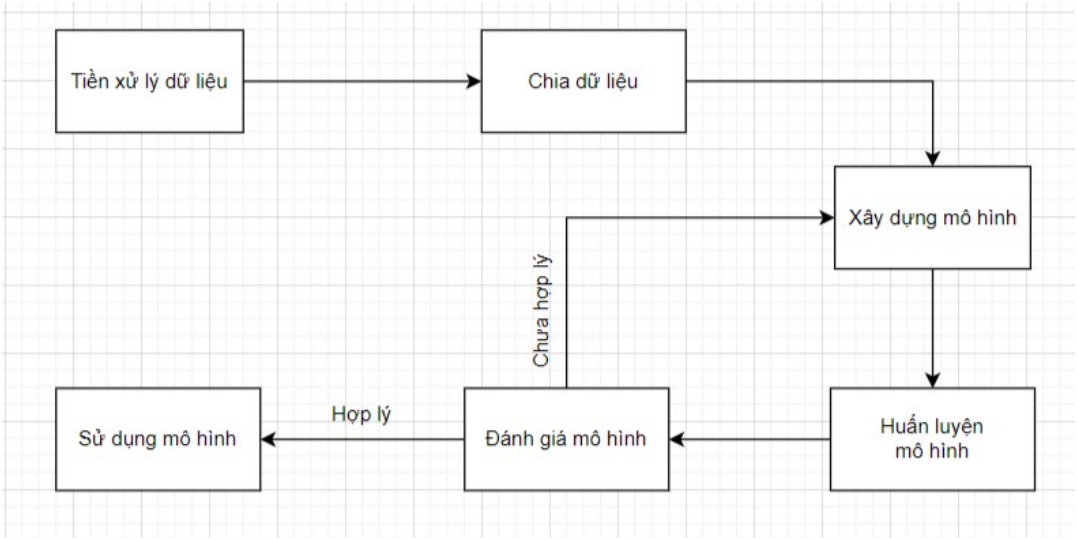
1.1 Tổng quan về bài toán phân lớp

1.1.1 Định nghĩa

Bài toán phân lớp (Classification) là một loại bài toán học máy, thuộc nhóm các bài toán học có giám sát (Supervised Learning), trong đó mục tiêu là xác định nhãn của một đối tượng hoặc nhóm đối tượng dựa trên các thuộc tính hoặc đặc điểm của nó. Bằng cách sử dụng dữ liệu huấn luyện, một mô hình phân lớp được xây dựng để tìm ra quan hệ giữa các thuộc tính và nhãn, sau đó sử dụng quan hệ đó để dự đoán nhãn của một đối tượng mới dựa trên các thuộc tính của nó.

Ví dụ 1.1.1. *Một ví dụ về bài toán phân lớp là xác định xem một email có phải là spam hay không. Với bài toán này, ta có một tập dữ liệu gồm các email đã được gắn nhãn là spam hoặc không phải spam. Mỗi email có thể được mô tả bởi một tập hợp các đặc trưng như tiêu đề, nội dung, địa chỉ email của người gửi,... Mục tiêu của bài toán phân lớp là học một mô hình từ tập dữ liệu huấn luyện để có thể dự đoán xem một email mới có phải là spam hay không. Một khi đã có mô hình, ta có thể sử dụng nó để phân loại các email mới mà không cần phải đánh giá thủ công từng email.*

Dưới đây là một quy trình đầy đủ của một bài toán phân lớp:



Hình 1.1: Quy trình bài toán phân hớp

- **Tiền xử lý dữ liệu**: Xử lý dữ liệu đầu vào bằng cách loại bỏ hoặc chỉnh sửa các giá trị bị thiếu, chuẩn hoá dữ liệu...
- **Chia dữ liêu**: Chia tập dữ liệu thành hai phần để huấn luyện và kiểm thử mô hình, thường tỷ lệ là 7-3 (70% để huấn luyện và 30% để kiểm thử) hoặc 8-2.
- **Xây dựng mô hình**: Chọn mô hình phù hợp cho bài toán phân lớp, bao gồm lựa chọn thuật toán phân lớp, cấu hình mô hình, và các tham số tùy chọn.
- **Huấn luyện mô hình**: Sử dụng tập dữ liệu huấn luyện để huấn luyện mô hình vừa xây dựng được.
- **Đánh giá mô hình**: Sử dụng tập kiểm thử để đánh giá mô hình. Thông qua một số tiêu chí đánh giá, chúng ta sẽ quyết định xem mô hình có phù hợp không. Nếu phù hợp sẽ chuyển qua bước sử dụng mô hình, nếu chưa phù hợp thì sẽ thay đổi một số tham số số của thuật toán hoặc thậm chí thay đổi cả thuật toán.
- **Sử dụng mô hình**: Sau khi mô hình được huấn luyện và đánh giá, chúng ta có thể sử dụng nó để dự đoán nhãn cho các đối tượng mới chưa được gán nhãn.

1.1.2 Các dạng bài toán phân lớp

Phân lớp nhị phân

Phân lớp nhị phân (Binary Classification) là bài toán phân lớp với nhiệm vụ là phân loại các mẫu dữ liệu vào hai lớp. Thông thường, các bài toán phân lớp nhị phân liên quan đến một lớp là trạng thái bình thường và một lớp khác là trạng thái bất thường. Lớp cho trạng thái bình thường được gán nhãn lớp 0 và lớp có trạng thái bất thường được gán nhãn lớp 1.

Ví dụ 1.1.2. *Ví dụ trong bài toán phát hiện email rác (spam) thì "không phải là email rác" là trạng thái bình thường và "email rác" là trạng thái bất thường.*

Các thuật toán phổ biến cho bài toán phân lớp nhị phân gồm:

- Hồi quy logistic
- k-Nearest Neighbors
- Cây quyết định
- Support vector machine
- Naive Bayes

Phân lớp nhiều lớp

Phân lớp nhiều lớp (Multi-class Classification) là một bài toán phân loại dữ liệu thành nhiều lớp khác nhau. Không giống như phân lớp nhị phân, phân lớp nhiều lớp không có khái niệm về kết quả bình thường và bất thường. Thay vào đó, các mẫu được phân loại là thuộc về một trong một loạt các lớp đã biết. Số lượng nhãn lớp có thể rất lớn đối với một số bài toán.

Ví dụ 1.1.3. *Ví dụ trong bài toán phân loại hình ảnh, ta có thể phân loại các hình ảnh thành nhiều lớp như động vật, con người, phương tiện, vật dụng, vật cảnh,...*

Các thuật toán phổ biến cho bài toán phân lớp nhiều lớp gồm:

- Random forest
- k-Nearest Neighbors
- Cây quyết định
- Gradient boosting
- Naive Bayes

1.1.3 Các tiêu chí đánh giá

TN, TP, FN, FP là các chỉ số đánh giá trong bài toán phân lớp. Chúng được sử dụng để đo lường hiệu quả của một mô hình phân lớp bằng cách tính tỷ lệ cặp (input, output) được dự đoán đúng và sai.

TP (True Positive): Số lượng cặp (input, output) được dự đoán là positive và thực tế là positive.

TN (True Negative): Số lượng cặp (input, output) được dự đoán là negative và thực tế là negative.

FN (False Negative): Số lượng cặp (input, output) được dự đoán là negative mà thực tế là positive.

FP (False Positive): Số lượng cặp (input, output) được dự đoán là positive mà thực tế là negative.

Confusion Matrix

Confusion matrix (ma trận nhầm lẫn) là một công cụ phân tích mô hình máy học phân loại. Nó hiển thị số lượng các mẫu được phân loại đúng và sai trong từng lớp. Ma trận nhầm lẫn có hai chiều: dòng và cột. Dòng thể hiện các giá trị thực tế của mẫu, còn cột thể hiện các giá trị dự đoán của mô hình.

Ví dụ 1.1.4. *Ví dụ, nếu bạn có ba lớp A, B, và C và mô hình của bạn phân loại 100 mẫu, ma trận nhầm lẫn có thể có dạng như sau:*

	A	*B*	*C*
A	*50*	*5*	*0*
B	*2*	*40*	*3*
C	*1*	*8*	*20*

Trong đó số trong mỗi ô cho thấy số lượng các mẫu được phân loại thành lớp tương ứng so với giá trị thật. Ví dụ trên cho thấy rằng 50 mẫu thật sự thuộc lớp A được phân loại chính xác thành lớp A, nhưng 5 mẫu thật sự thuộc lớp A được phân loại sai thành lớp B.

Accuracy

Accuracy (độ chính xác): là một trong những chỉ số đánh giá hiệu quả của một mô hình phân lớp. Nó được tính bằng số lượng mẫu được dự đoán chính xác chia cho tổng số lượng mẫu. Tuy nhiên, trong một số trường hợp, độ chính xác có thể không phản ánh đầy đủ hiệu suất của mô hình, đặc biệt khi tỷ lệ phân phối giữa các lớp không đồng đều.

$$Accuracy = \frac{TN + TP}{N}$$

Recall

Recall được định nghĩa là tỉ lệ số điểm Positive mô hình dự đoán đúng trên tổng số điểm thật sự là Positive (hay tổng số điểm được gán nhãn là Positive ban đầu). Tỷ lệ này càng cao thì cho thấy khả năng bỏ sót các điểm Positive là thấp.

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

Precision

Precision được định nghĩa là tỉ lệ số điểm Positive mô hình dự đoán đúng trên tổng số điểm mô hình dự đoán là Positive. Số này càng cao thì mô hình nhận các điểm Positive càng chuẩn.

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

F1 Score

Như vậy chúng ta đã có 2 khái niệm Precision và Recall và mong muốn 2 thằng này càng cao càng tốt. Tuy nhiên trong thực tế nếu ta điều chỉnh model để tăng Recall quá mức có thể dẫn đến Precision giảm và ngược lại, cố điều chỉnh model để tăng Precision có thể làm giảm Recall. Nhiệm vụ của chúng ta là phải cân bằng 2 đại lượng này. F1 Score có giá trị nằm trong khoảng từ 0 đến 1, với giá trị càng gần 1 thể hiện mô hình có hiệu suất càng tốt.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

ROC - AUC

TPR(True Positive Rate/Sentivity/Recall): Biểu diễn tỷ lệ phân loại chính xác các mẫu Positive trên tất cả các mẫu Positive.

Specificity: Biểu diễn tỷ lệ phân loại chính xác các mẫu Negative trên tất cả các mâu Negative, được tính theo công thức:

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

FPR(False Positive Rate/Fall-out): Biểu diễn tỷ lệ gắn nhãn sai các mẫu Negative thành Positive trên tất cả các mẫu Negative, được tính theo công thức:

$$FPR = 1 - Specificity$$

Một số tiêu chí đánh giá khác đối với bài toán phân lớp nhiều lớp:

Đối với các bài toán phân lớp nhiều lớp chúng ta còn có các cách đánh giá khác như sau:

- **Macro-average**: là trung bình của các chỉ số (có thể là precision, recall hoặc F1-score...). Nó tính toán chỉ số phân lớp cho từng lớp rồi tính trung bình cho tất cả các lớp.

  Ví dụ 1.1.5. *Với bài toán phân lớp có 3 lớp, sau khi đánh giá bài toán, ta được chỉ số precision của từng lớp là : 0.7, 0.8, 0.9. Khi đó chỉ số Macro-average precision sẽ được tính như sau:*

$$\textit{Macro avg precision} = \frac{0.7+0.8+0.9}{3} = 0.8$$

- **Weighted-average**: là trung bình có trọng số của các chỉ số (có thể là precision, recall hoặc F1-score...).

  Ví dụ 1.1.6. *Với bài toán phân lớp có 3 lớp, tổng số quan sát là 10, số quan sát của mỗi lớp lần lượt là 5,3,2. Sau khi đánh giá bài toán, ta được chỉ số precision của từng lớp là : 0.7, 0.8, 0.9. Khi đó chỉ số Weighted-average precision sẽ được tính như sau:*

$$\textit{Macro avg precision} = \frac{0.7\times 5+0.8\times 3+0.9\times 2}{10} = 0.77$$

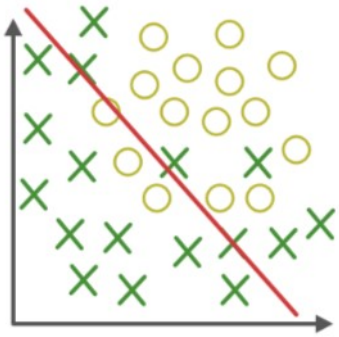
1.1.4 Một số vấn đề trong bài toán phân lớp

Độ lệch

Độ lệch (bias) là chỉ số đo lường sự khác biệt giữa giá trị trung bình của mô hình dự đoán và giá trị thật sự của dữ liệu. Nó thường được hiểu là sự sai lệch của mô hình dự đoán so với giá trị thật sự. Một mô hình có độ lệch cao sẽ dự đoán sai xa giá trị thật sự, còn một mô hình có độ lệch thấp sẽ dự đoán gần giá trị thật sự hơn. Nguyên

nhân của sự sai lệch thường là do mô hình quá đơn giản trong khi dữ liệu có mối quan hệ phức tạp hơn và thậm chí nằm ngoài khả năng biểu diễn của mô hình. Vì vậy trong tình huống này để giảm bớt độ chệch thì chúng ta thường sử dụng mô hình phức tạp hơn để tận dụng khả năng biểu diễn tốt hơn của chúng trên những tập dữ liệu kích thước lớn.

Khi độ lệch cao nghĩa là có sự khác biệt lớn giữa giá trị mô hình dự đoán và giá trị thật sự, điều này sẽ dẫn đến hiện tượng underfitting. Underfitting là tình trạng khi mô hình học máy không đủ mạnh để hợp lý diễn giải dữ liệu huấn luyện. 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. Underfitting có thể xảy ra khi mô hình quá đơn giản, khi không có đủ dữ liệu huấn luyện hoặc khi các thông số mô hình không được điều chỉnh đúng cách.



Hình 1.2: Underfitting

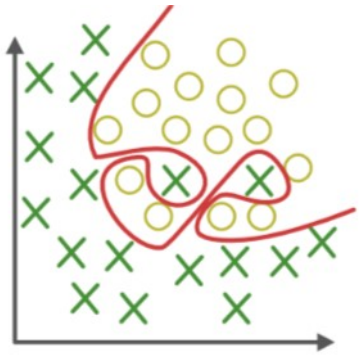
Trong huấn luyện mô hình, việc giảm độ lệch có thể được thực hiện bằng cách sử dụng các phương pháp học có giảm độ lệch như học tăng cường, sử dụng các thuật toán regularization, hay sử dụng các phương pháp ensemble learning.

Phương sai

Phương sai (variance) là một thước đo để đo lường mức độ phân tán của dữ liệu. Nó đo lường sự biến thiên của dữ liệu so với giá trị trung bình. Phương sai càng lớn thì dữ liệu càng phân tán rộng, còn phương sai càng nhỏ thì dữ liệu càng tập trung gần

với giá trị trung bình. Phương sai là một thước quan trọng để đánh giá độ chính xác của một mô hình dự báo.

Phương sai cao nghĩa là mô hình của bạn dự báo ra giá trị có mức độ dao động lớn nhưng thiếu tổng quát. Yếu tố thiếu tổng quát được thể hiện qua việc giá trị dự báo có thể khớp tốt mọi điểm trên tập huấn luyện nhưng rất dao động xung quanh giá trị thật sự trên tập huấn luyện. Phương sai cao dẫn đến hượng tượng overfitting, mô hình có khả năng tạo ra kết quả dự báo rất tốt trên tập dữ liệu huấn luyện nhưng không tốt trên tập dữ liệu chưa được huấn luyện. Những lớp mô hình phức tạp được huấn luyện trên tập huấn luyện nhỏ thường xảy ra hiện tượng phương sai cao và dẫn tới việc học giả mạo thông qua bắt chước dữ liệu hơn là học qui luật tổng quát.



Hình 1.3: Overfitting

Trong huấn luyện mô hình, việc giảm độ lệch có thể được thực hiện bằng cách sử dụng các phương pháp học có giảm độ lệch như học tăng cường, sử dụng các thuật toán regularization, hay sử dụng các phương pháp ensemble learning.

Mục tiêu của chúng ta là tìm một mô hình sao cho có phương sai và độ lệch đều thấp.

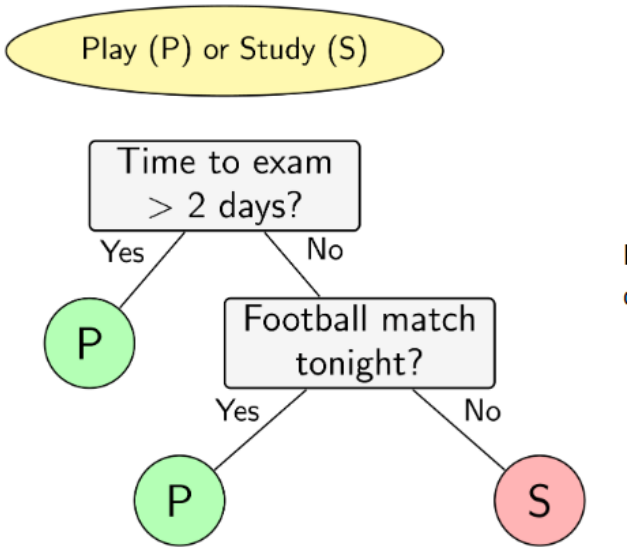
1.2 Thuật toán cây quyết định

Có rất nhiều mô hình phân lớp dữ liệu đã được các nhà toán tìm ra và áp dụng trong nhiều lĩnh vực khác nhau, có thể kể đến một số thuật toán phân lớp như mạng

nơ-ron, mô hình hồi quy logistic, cây quyết định... Trong số những mô hình đó, cây quyết định với những ưu điểm của mình được đánh giá là một công cụ mạnh, phổ biến và đặc biệt thích hợp cho bài toán phân lớp. Cây quyết định có nhiều ưu điểm như: xây dựng tương đối nhanh; đơn giản, dễ hiểu. Hơn nữa các cây có thể dễ dàng được chuyển đổi sang các câu lệnh SQL để có thể được sử dụng để truy nhập cơ sở dữ liệu một cách hiệu quả. Cuối cùng, việc phân lớp dựa trên cây quyết định đạt được sự tương tự và đôi khi là chính xác hơn so với các phương pháp phân lớp khác.

1.2.1 Định nghĩa

Cây quyết định (decision tree) là một phương pháp học máy với cấu trúc dạng cây sử dụng để phân tích và dự đoán dữ liệu, đây là một trong những hình thức mô tả dữ liệu trực quan nhất, dễ hiểu nhất đối với người dùng. Cấu trúc của một cây quyết định bao gồm các nút và các nhánh. Nút dưới cùng được gọi là nút lá, trong mô hình phân lớp dữ liệu chính là các giá trị của các nhãn. Các nút khác nút lá được gọi là các nút con, đây còn là các thuộc tính của tập dữ liệu. Nút đầu tiên được gọi là nút gốc của cây.



Hình 1.4: Minh hoạ cây quyết định

1.2.2 Cách khởi tạo cây quyết định

Đối với những bộ dữ liệu có số lượng biến đầu vào p lớn, việc lựa chọn nút nào làm gốc sẽ khá khó khăn. Ta không thể lựa chọn ngẫu nhiên 1 nút gốc bất kỳ vì xác suất để lựa chọn đó là tối ưu chỉ là $\frac{1}{p}$ rất nhỏ. Do đó ta cần phải có tiêu chí để lựa chọn biến cho phù hợp. Hai chỉ số entropy và gini sẽ giúp chúng ta làm điều này.

Tinh khiết và vẩn đục

Giả sử rằng tại một nút cha ta có 100 quan sát cần phải phân vào các nút con. Có 3 phương án lựa nút con theo tương ứng với 3 biến, kết quả thu được như sau:

- Biến x: 50 nhãn 1, 50 nhãn 0.
- Biến y: 70 nhãn 1, 30 nhãn 0.
- Biến z: 100 nhãn 1, 0 nhãn 0.

Nhận thấy nếu ta lựa chọn biến z thì sẽ tối ưu vì chúng ta không cần phải phân chia tiếp nữa, ta có thể coi nút này là một nút lá. Trong khi đó nếu chọn biến x để phân chia thì dường như là vô nghĩa, vì tỷ lệ các nhãn là như nhau, không có tác dụng phân tách.

Như vậy mục tiêu của chúng ta là lựa chọn biến sao cho sau khi phân chia thì kết quả trả về tại nút con chỉ thuộc về một lớp, trong trường hợp này ta gọi là tinh khiết (purity). Trái ngược lại với tinh khiết sẽ là khái niệm vẩn đục (impurity), tức phân phối của các nhãn tại nút con còn khá mập mờ, không có xu hướng thiên về một nhãn nào cụ thể. Nếu ra quyết định phân loại dựa trên kịch bản dẫn tới nút lá sẽ trả về kết quả không đáng tin cậy.

Entropy

Entropy là một khái niệm khoa học được sử dụng lần đầu tiên trong nhiệt động lực học và sau đó được phổ biến trong các lĩnh vực khác như vật lý, hoá học, y sinh, lý

thuyết thông tin,... Trong thuật toán cây quyết định chúng ta sẽ sử dụng Entropy để đánh giá mức độ tinh khiết của phân phối xác suất của một sự kiện.

Cho một phân phối xác suất của một biến rời rạc x có thể nhận n giá trị khác nhau $x_1, x_2, ..., x_n$. Giả sử rằng xác suất để $x = x_i$ là p_i với $\sum_{i=1}^{n} p_i = 1$. Ký hiệu phân phối này là $\boldsymbol{p} = (p_1, p_2, ..., p_n)$. Entropy của phân phối này được tính như sau:

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

Ví dụ 1.2.1. *Với n=2, trong trường hợp $\boldsymbol{p}$ là tinh khiết nhất, nghĩa là 1 trong 2 giá trị p_i bằng 1, giá trị còn lại bằng 0. Khi đó entropy của phân phối này là $H(\boldsymbol{p})$. Ngược lại, khi $\boldsymbol{p}$ vẩn đục nhất là khi cả hai giá trị p_i đề bằng 0.5, khi đó hàm entropy đạt giá trị lớn nhất.*

Thuật toán cây quyết định sẽ sử dụng tổng có trọng số của entropy tại các nút lá sau khi xây dựng cây quyết định làm hàm mất mát. Trọng số ở đây được lấy theo tỷ lệ phần trăm quan sát trên từng nút lá. Điều đó có nghĩa rằng với những nút lá có số lượng quan sát lớn thì ảnh hưởng của nó lên hàm mất mát là lớn hơn so với những nút lá có số lượng quan sát nhỏ. Việc này là hợp lý vì việc phân loại sai những nút lá lớn gây hậu quả nghiêm trọng hơn so với phân loại sai nút nhỏ. Để tối thiểu hoá hàm mất mát thì chúng ta phải lựa chọn biến và ngưỡng sao cho tổng giá trị của hàm mất mát là nhỏ nhất. Như vậy ta phải tìm các cách phân chia hợp lý (thứ tự chọn thuộc tính hợp lý) sao cho hàm mất mát cuối cùng đạt giá trị càng nhỏ càng tốt. Việc này đạt được bằng cách chọn ra thuộc tính sao cho nếu dùng thuộc tính đó để phân chia, entropy tại mỗi bước giảm đi một lượng lớn nhất.

Xét bài toán gồm C nhãn khác nhau. Ta sẽ làm việc với một nút lá bất kỳ. Xét tập S là tập dữ liệu của một nốt lá, kích thước của tập này là N. Giả sử rằng trong N điểm dữ liệu này số điểm dữ liệu có nhãn i là N_i. Khi đó, entropy tại nút này được tính bởi:

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

Giả sử thuộc tính được chọn là x. Dựa trên x, các điểm dữ liệu trong S được chia thành K nút $S_1, S_1, ..., S_K$ và kích thước của mỗi nút là $m_1, m_2, ..., m_K$. Khi đó tổng có

trọng số entropy được tính như sau:

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

Ta sẽ lựa chọn thuộc tính x sao cho tổng có trọng số của entropy đạt giá trị nhỏ nhất.

Gini index

Chỉ số Gini index là một lựa chọn khác bên cạnh hàm entropy được sử dụng để đo lường mức độ bất bình đẳng trong phân phối của các lớp. Để tính gini index, đầu tiên ta đi tính chỉ số gini ở các nút.

$$gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Sau khi tính được chỉ số gini ở nút cha và các nút con. Ta tính được chỉ số gini index:

$$gini\_index = gini(p) - \sum_{i=1}^{K} \frac{m_k}{M} gini(c_k)$$

Trong đó gini(p) là chỉ số gini ở nút cha, K là số lượng nút con được tách ra từ nút cha, $gini(c_k)$ là chỉ số gini ở nút con thứ k. M là số phần tử nút cha, m_i là số phần tử ở nút con thứ i.

Mục tiêu là tách sao cho chỉ số gini ở các nút con nhỏ, hay gini index mong muốn càng lớn càng tốt.

1.2.3 Điều kiện dừng

- Nếu nút đó có entropy bằng 0, tức mọi điểm trong nút đều thuộc một lớp.
- Nếu nút đó có số phần tử nhỏ hơn một ngưỡng nào đó. Trong trường hợp này ta chấp nhận có một số điểm bị phân lớp sai để để tránh overfitting. Lớp cho nút lá này có thể được xác định dựa trên lớp chiếm đa số trong nút.

- Nếu khoảng cách từ nút đó đến nút gốc đạt tới một giá trị nào đó. Việc hạn chế chiều sâu của cây này làm giảm độ phức tạp của cây và phần nào đó giúp tránh overfitting.
- Nếu tổng số nút vượt quá một ngưỡng nào đó.
- Nếu việc phân chia nút đó không làm giảm entropy qua nhiều.

Chương 2

Phương pháp Bagging

2.1 Ensemble Learning

2.1.1 Giới thiệu

Trong một vài thập kỷ gần đây, các hệ thống Ensemble ngày càng được chú ý trong cộng đồng học máy. Ensemble Learning là một kỹ thuật quan trọng trong học máy, nó kết hợp một số mô hình cơ sở để tạo ra một mô hình tối ưu. Ý tưởng của việc các kết hợp các mô hình xuất phát từ một ý tưởng hợp lý là: các mô hình khác nhau có khả năng khác nhau, có thể thực hiện tốt nhất các loại công việc khác nhau, khi kết hợp các mô hình này với nhau một cách hợp lý thì sẽ tạo thành một mô hình kết hợp mạnh mẽ có khả năng cải thiện hiệu suất tổng thể so với việc chỉ dùng các mô hình một cách đơn lẻ.

Ban đầu Ensemble Learning được phát triển để giảm phương sai nhằm cải thiện độ chính xác của một hệ thống ra quyết định tự động, các hệ thống Ensemble đã được sử dụng thành công để giải quyết nhiều vấn đề học máy, chẳng hạn như lựa chọn tính năng, ước tính độ tin cậy, mất cân bằng dữ liệu...

2.1.2 Ensemble Learning là gì?

Ensemble Learning là một kỹ thuật trong học máy, nó sử dụng nhiều mô hình của máy học để tạo ra một mô hình mạnh hơn so với mỗi mô hình đơn lẻ. Nó kết hợp các mô hình con để cải thiện độ chính xác so với một mô hình duy nhất.

Khi bạn sử dụng một mô hình để giải quyết một bài toán nhưng đầu ra khi chạy mô hình đó không tốt nên bạn thử các mô hình khác. Sau khi tìm được mô hình phù hợp, bạn lại phải chỉnh chỉnh sửa sửa từ thuật toán đến các tham số để mô hình đạt độ chính xác cao nhất. Tất cả những việc kể trên sẽ ngốn của bạn một đống thời gian bởi bạn phải chạy từng mô hình một, thế nên để nhanh hơn bạn kết hợp những mô hình "học yếu" này lại để tạo ra một mô hình "học mạnh". Đó chính là Ensemble Learning.

2.1.3 Phân loại

Về cơ bản, Ensemble Learning có thể được chia thành 3 loại chính: bagging, boosting và stacking.

- **Bagging**: Là viết tắt của Bootstrapped Aggregating, phương pháp này xây dựng một lượng lớn các mô hình (thường là cùng loại) trên những mẫu dữ liệu (được lấy theo phương pháp bootstrap từ bộ dữ liệu ban đầu, phương pháp này sẽ được trình bày ở phần 2.2), những mô hình này sẽ được huấn luyện độc lập và song song với nhau. Sau đó dùng các phương pháp tổng hợp các mô hình con này lại (sử dụng các phương pháp Aggregating được trình bày ở phần 2.3) để cho ra kết quả cuối cùng.

- **Boosting**: Xây dựng một lượng lớn các mô hình (thường là cùng loại). Mỗi mô hình sau 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 nhật qua mỗi mô hình (cụ thể ở đây là trọng số của những dữ liệu dự đoán đúng sẽ không đổi, còn trọng số của những dữ liệu dự đoán sai sẽ được tăng thêm). Chúng ta sẽ lấy kết quả của mô hình cuối

cùng trong chuỗi mô hình này làm kết quả trả về (vì mô hình sau sẽ tốt hơn mô hình trước nên tương tự kết quả sau cũng sẽ tốt hơn kết quả trước).

- **Stacking**: Stacking yêu cầu ít nhất hai cấp mô hình. Ở cấp mô hình thứ nhất, nhiều mô hình cơ bản được huấn luyện trên tập dữ liệu đào tạo và tạo ra các dự đoán. Ở cấp mô hình thứ hai, một mô hình meta được huấn luyện bằng cách sử dụng các dự đoán từ các mô hình cơ bản làm đầu vào, sau đó meta mô hình 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 mô hình giảm variance. Còn Boosting và Stacking tập trung vào việc giảm bias

2.2 Phương pháp lấy mẫu bootstrap

Bootstrap là một phương pháp lấy mẫu trong thống kê và học máy. Nó được sử dụng để tạo ra mẫu từ tập dữ liệu ban đầu bằng cách lặp lại việc chọn ngẫu nhiên một phần tử trong tập dữ liệu và chọn nó vào mẫu mới. Trong khi các phần tử có thể được chọn nhiều lần, có thể còn một số phần tử không được chọn vào mẫu. Việc lặp đi lặp lại sẽ tiếp tục cho đến khi tập dữ liệu mẫu đạt đến số lượng mẫu tương đương với tập dữ liệu gốc. Điều này giúp cho mẫu có đầy đủ thông tin về tập dữ liệu gốc. Phương pháp bootstrap được sử dụng rộng rãi trong mô hình học máy và trong các phân tích thống kê.

Ví dụ 2.2.1. *Giả sử có 5 quan sát được dán nhãn A, B, C, D và E trên 5 quả bóng và bỏ tất cả chúng vào trong 1 cái giỏ. Ta sẽ thực hiện việc lấy mẫu bootstrap với kích thước bằng với kích thước ban đầu của mẫu. Từ 5 quả bóng này, ta lấy một quả bóng từ giỏ một cách ngẫu nhiên và ghi lại nhãn của chúng, sau đó bỏ lại quả bóng vừa bốc được vào giỏ và tiếp tục lấy ra một quả bóng một cách ngẫu nhiên, ghi lại nhãn của quả bóng và bỏ lại quả bóng vào trong giỏ, tiếp tục thực hiện việc lấy mẫu như vậy cho tới khi kết thúc. Kết quả của việc lấy mẫu như trên có thể như sau:*

C, D, E, E, A

Các quan sát có thể lặp lại trong mẫu bootstrap (quan sát E lặp lại 2 lần), một số quan sát có thể không xuất hiện trong mẫu bootstrap (quan sát B). Đây là đặc trưng của phương pháp lấy mẫu bootstrap.

2.3 Aggregating

Aggregating (tổng hợp) là một quá trình trong Bagging để tạo ra dự đoán cuối cùng. Trong Bagging, mỗi mô hình con được huấn luyện sẽ tạo ra một dự đoán cho mỗi điểm dữ liệu. Khi tất cả các mô hình đã hoàn tất việc huấn luyện, quá trình tổng hợp sẽ sử dụng các dự đoán đó để tạo ra dự đoán cuối cùng cho mỗi điểm dữ liệu.

Phương pháp tổng hợp phổ biến nhất là trung bình đồng của các dự đoán từ các mô hình. Tuy nhiên, trong một số trường hợp, phương pháp tổng hợp khác có thể được sử dụng, như trung vị hoặc phần tử xuất hiện nhiều nhất. Quá trình tổng hợp của Bagging đảm bảo rằng mô hình dự đoán cuối cùng sẽ giảm variance so với mỗi mô hình độc lập, tăng tính chính xác của dự đoán.

Một số phương pháp tổng hợp:

- **Majority voting**: Là một phương pháp tổng hợp trong Ensemble learning, đặc biệt là trong Bagging. Đối với mỗi điểm dữ liệu, nó sẽ tạo ra nhiều mô hình phân loại , sau đó chọn mô hình có nhãn dự đoán nhiều nhất cho điểm dữ liệu đó. Nó tạo ra một "bộ đếm" cho từng nhãn của từng mô hình và chọn nhãn có số lần xuất hiện nhiều nhất làm nhãn dự đoán cho điểm dữ liệu.

- **Average**: Phương pháp average trong Bagging là một cách tính trung bình các kết quả dự đoán từ nhiều mô hình con. Với mỗi mô hình con, chúng ta sẽ dự đoán kết quả cho mỗi mục tiêu và sau đó tính trung bình các kết quả dự đoán đó. Kết quả cuối cùng sẽ là giá trị trung bình của các kết quả dự đoán từ tất cả các mô hình con.

Tại sao Aggregating trong Bagging lại tốt? Vì mỗi mô hình được huấn luyện trên một tập dữ liệu khác nhau, nên sẽ không có 2 mô hình nào giống nhau hoặc có sự tương quan cao giữa chúng. Điều này sẽ giúp cho kết quả tổng hợp từ các mô hình sẽ có độ phức tạp thấp hơn và phân bố chuẩn hơn so với mô hình đơn. Từ đó, phương sai trong kết quả dự báo sẽ được giảm và chất lượng dự báo của mô hình cũng sẽ tốt hơn.

2.4 Out of bag

Xét mẫu có N quan sát:

- Lấy ngẫu nhiên 1 quan sát, xác suất để 1 quan bất kỳ được chọn là: $\frac{1}{N}$
- Lấy ngẫu nhiên lần lượt N quan sát, xác suất để 1 quan sát bất kỳ không được chọn là: $(1-\frac{1}{N})^N$
- Khi N đủ lớn thì giá trị trên tiến tới $\frac{1}{e}$

Do đó, xét mẫu có N quan sát, nếu dùng phương pháp Bootstrap để lấy một mẫu mới có N quan sát thì:

- Mỗi mẫu bootstrap chứa xấp xỉ 63.2% số lượng dữ liệu trong tệp ban đầu
- Số lượng mẫu còn lại khoảng 36.8% gọi là out of bag, lượng này dùng để kiểm thử.

Out-of-Bag (OOB) là một khái niệm quan trọng trong phương pháp Bagging. Trong mỗi lần lấy mẫu bootstrap, một số dữ liệu sẽ không được chọn làm mẫu để huấn luyện mô hình sẽ được dùng để kiểm thử mô hình, vì vậy chúng ta gọi chúng là dữ liệu Out-of-Bag (OOB).

Dữ liệu OOB có thể được sử dụng để đánh giá chất lượng của mô hình. Vì mô hình đã huấn luyện trên dữ liệu khác nên nó sẽ có khả năng đánh giá tốt hơn trên dữ liệu mới và chưa biết, do đó việc sử dụng dữ liệu OOB để đánh giá chất lượng mô hình là một trong những ưu điểm của phương pháp Bagging.

2.5 Thuật toán Bagging

Như đã nêu ở trên, Bagging là phương pháp kết hợp nhiều mô hình học máy có độ chính xác cao nhưng có phương sai lớn để tăng tính ổn định. Trước khi có Bagging, mô hình cây quyết định phải sử dụng các kỹ thuật cắt tỉa cây để tăng tính ổn định (giảm hiện tượng overfitting) nhưng phải đánh đổi bằng việc giảm độ chính xác.

Phương pháp này được xem như là một phương pháp tổng hợp két quả có được từ các mẫu bootstrap. Ý tưởng chính của phương pháp này như sau: Cho một tập huấn luyện $D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ và giả sử chúng ta muốn phân lớp quan sát x

- Một mẫu gồm B tập dữ liệu, mỗi tập dữ liệu gôm N quan sát được lấy từ tập gốc D theo phương pháp bootstrap.
- Sử dụng mô hình phân lớp đối với mỗi tập D_b (b=1, 2,..., B) và lần được thu thập các kết quả dự báo trên mỗi tập D_b
- Kết quả tổng hợp cuối cuồi được tính toán thông qua phương pháp tổng hợp Aggregating.

Algorithm 1 Thuật toán Bagging$(\mathscr{A}, D, J)$:

1: Xét bộ dữ liệu gốc $D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, với $x_i = (x_{i,1}, x_{i,2}, ..., x_{i,p})^T$

2: **for** $j = 1, J$ **do**

3: Lấy mẫu bootstrap D_j với kích thước N từ tập D

4: Chạy thuật toán $\mathscr{A}$ trên bộ dữ liệu này để huấn luyện mô hình

$$h_j = \mathscr{A}(D_j)$$

5: **end for**

6: Xây dựng mô hình tổng hợp. Với mô hình phân lớp, sử dụng h_i để bỏ phiếu chọn ra phân lớp có nhiều mô hình chọn nhất :

$$h(\mathbf{x}) = \arg\max_c \sum_{j=1}^{J} \mathbb{I}(h_j(\mathbf{x}) = c)$$

2.6 Thuật toán Random Forest

2.6.1 Định nghĩa

Random Forest là một phương pháp học máy đồng thời sử dụng cả phương pháp Bagging và Decision Tree. Nó sử dụng các cây quyết định riêng biệt (Decision Tree) để dự đoán và sau đó áp dụng phương pháp Bagging để tạo ra một tập hợp các cây quyết định và sau đó tổng hợp kết quả của các cây đó để đưa ra quyết định cuối cùng.

Mô hình rừng ngẫu nhiên sử dụng Bagging đối với thuật toán cây quyết định với một thay đổi nhỏ. Trong đó, khi xây dựng cây quyết định từ tập mẫu D_i, tại mỗi bước phân chia nút trong cây, ta chỉ chọn ngẫu nhiên m thuộc tính trong p thuộc tính $A_1, A_2, \ldots, A_p$ để chọn ra thuộc tính phân chia tốt nhất.

Mã giả cho thuật toán rừng ngẫu nhiên:

Algorithm 2 Thuật toán RandomForest(D, J, m)

1: Xét bộ dữ liệu gốc $D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, với $x_i = (x_{i,1}, x_{i,2}, ..., x_{i,p})^T$

2: **for** $j = 1, J$ **do**

3: Lấy mẫu bootstrap D_j với kích thước N từ tập D

4: Xây dựng cây quyết định $h_j = T_j$ trên tập D_j sao cho

- Tại mỗi lần phân chia, chọn ngẫu nhiên m thuộc tính trong p thuộc tính dữ liệu.
- Tìm thuộc tính phân chia tốt nhất (theo entropy, gini...)
- Phân chia đến khi mỗi nút có không quá n_{min} mẫu dữ liệu

5: **end for**

6: Xây dựng mô hình tổng hợp như Bagging :

$$h(\mathbf{x}) = \arg\max_{c} \sum_{j=1}^{J} \mathbb{I}(h_j(\mathbf{x}) = c)$$

2.6.2 Đánh giá bằng out of bag

Đánh giá bằng Out-of-bag (OOB) là một phương pháp đánh giá hiệu quả của mô hình Random Forest mà không cần sử dụng tập dữ liệu kiểm tra riêng. Mỗi cây trong Random Forest được xây dựng trên một tập dữ liệu bootstrap khác nhau, vì vậy một số mẫu trong tập dữ liệu huấn luyện sẽ không được chọn trong tập dữ liệu cho mỗi cây, những mẫu này có thể được sử dụng để đánh giá hiệu quả của mô hình.

Để đánh giá mô hình bằng OOB, ta sẽ dùng những mẫu OOB để dự đoán nhãn của mô hình, và so sánh với nhãn thật sự của chúng. Điểm OOB là tỷ lệ mẫu được dự đoán đúng trên tổng số mẫu OOB. Đánh giá bằng OOB được coi là một phương pháp đánh giá tốt vì nó không yêu cầu một tập dữ liệu kiểm thử riêng biệt, và có thể giúp cho ta biết được sự tương quan giữa các biến và tổng quan về mô hình.

Với mỗi mẫu dữ liệu $z_i = (x_i, y_i)$, sử dụng tất cả các cây được xây dựng không sử

dụng mẫu dữ liệu này để tính toán dự đoán $\widehat{h}(x_i)$. Ta xét với bài toán phân lớp:

$$\widehat{h}(x_i) = \arg\max_{c} \sum_{h_i \text{ không sử dụng mẫu } z_i} \mathbb{I}(h_i(x) = c)$$

2.6.3 Độ quan trọng của thuộc tính

Trong Random Forest, biến quan trọng (variable importance) được sử dụng để đo lường sự quan trọng của từng biến trong quá trình dự đoán. Độ quan trọng của mỗi biến được xác định bằng cách tính sự giảm thiểu độ chênh lệch giữa dự đoán thực tế và dự đoán mẫu khi biến đó bị loại bỏ. Biến có độ quan trọng cao sẽ có mức giảm thiểu độ chênh lệch cao, tức là nó có nhiều ảnh hưởng đến dự đoán. Còn biến có độ quan trọng thấp sẽ có mức giảm thiểu độ chênh lệch thấp, tức là nó có ít ảnh hưởng đến dự đoán.

Việc tính toán nhằm xác định được các thuộc tính quan trọng trong thuật toán rừng ngẫu nhiên cũng tương tự như việc sử dụng OOB để tính toán lỗi trong rừng ngẫu nhiên. Cách thực hiện như sau: Giả sử ta cần xác định độ quan trọng của thuộc tính thứ m, đầu tiên ta tính lỗi OOB, sau đó hoán vị ngẫu nhiên các giá trị của thuộc tính m trong tập dữ liệu OOB, lần lượt gửi các giá trị này xuống cây và đếm số các dự đoán đúng.

Các bước để tính toán độ quan trọng của thuộc tính được thực hiện như sau:

Algorithm 3 Thuật toán

1: Để tính độ quan trọng của thuộc tính k, for k= 1 to p:

2: **for** $i = 1, N$ **do**

Đặt $G_i = \{j : (x_i, y_i) \notin D_j\}$, J_i là kích thước của G_i

Tính $\hat{y}_{i,j} = \hat{h}_j(x_i)$ với mỗi $j \in G_i$

3: **end for**

4: **for** $i = 1, J$ **do**

Lấy mẫu bootstrap D_j với kích thước N từ tập D

Đặt $V_j = \{i : (x_i, y_i) \notin D_j\}$

Hoán đổi các giá trị của biến k $\{x_i : i \in V_j\}$, $\mathrm{Q_j} = \{x_i{}^* : i \in V_j\}$

Tính $\hat{y}^*_{i,j} = \hat{h}^*_j(x_i)$ với mỗi $j \in V_j$

5: **end for**

6: **for** $i = 1, N$ **do**

$$Imp_i = \frac{1}{J_i} \sum_{j \in G_i} I(y_i \neq \hat{y}^*_{i,j}) - \frac{1}{J_i} \sum_{j \in G_i} I(y_i \neq \hat{y}_{i,j})$$

7: **end for**

2.6.4 Các tham số của thuật toán Random Forest

Dưới đây là một số tham số quan trọng của thuật toán Random Forest:

n_estimators

n_estimators là một tham số cho biết số cây quyết định trong mô hình Random Forest. Tùy thuộc vào tập dữ liệu và bài toán phân lớp, số cây quyết định cần thiết có thể thay đổi. Nếu *n_estimators* quá nhỏ, mô hình có thể sẽ underfitting và không có khả năng phân lớp tốt. Nếu *n_estimators* quá lớn, mô hình có thể overfitting và làm tăng thời gian huấn luyện.

max_features

Trong Random Forest, mỗi cây chỉ chọn một tập nhỏ các thuộc tính trong quá trình xây dựng (bước ngẫu nhiên thứ 2), cơ chế này làm cho thuật toán thực thi với tập dữ liệu có số lượng thuộc tính lớn trong thời gian chấp nhận được khi tính toán. Nếu *max_features* quá cao sẽ làm tăng thời gian chạy và làm giảm tính đa dạng của từng cây đơn lẻ, có thể dẫn đến overfitting .Tuy nhiên nếu tham số này quá thấp có thể dẫn đến underfitting.

Nếu dữ liệu ban đầu của q thuộc tính thì:

- Với bài toán phân loại: dùng $\sqrt{q}$ thuộc tính
- Với bài toán hồi quy: dùng $\frac{q}{3}$ thuộc tính

max_depth

Là độ sâu tối đa của các cây quyết định trong mô hình. Nếu *max_depth* được thiết lập, các cây sẽ dừng phát triển khi đạt đến giá trị *max_depth* đó.

min_samples_split

Là số lượng quan sát tối thiểu cần phải có trong mỗi nút của cây quyết định để tiếp tục chia nhánh. 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.

min_samples_leaf

Là số lượng quan sát tối thiểu cần có trong mỗi lá của mỗi cây quyết định.

2.6.5 Tối ưu tham số cho thuật toán Random Forest

Các tham số sẽ ảnh hưởng lớn đến kết quả của bài toán. Việc tìm kiếm và chọn lựa các giá trị tối ưu cho các tham số trong mô hình Random Forest để cải thiện hiệu suất

của mô hình là thực sự cần. Tuy nhiên, tuỳ vào mỗi bộ dữ liệu và yêu cầu bài toán ta sẽ những một bộ tham số tối ưu khác nhau. Dưới đây em xin trình bày một số cách phổ biến để tìm ra bộ tham số phù hợp cho thuật toán Random Forest.

Grid Search

Grid Search là một kỹ thuật giúp tìm kiếm tham số phù hợp cho mô hình. Phương pháp này tìm kiếm tham số bằng cách lặp lại huấn luyện và đánh giá mô hình với các giá trị tham số khác nhau trên một lưới (grid) giá trị.

Ví dụ 2.6.1. *Ví dụ ta cần tối ưu 2 tham số n_estimators và max_features cho thuật toán Random Forest. Đầu tiên ta sẽ tạo một lưới các giá trị cho 2 tham số này như sau:grid={n_estimators= [2,3,4,5,6], max_features= [$\frac{q}{3}, \sqrt{q}$]}. Sau đó huấn luyện thuật toán với từng cặp giá trị tham số được lựa chọn trong lưới trên (kết hợp mỗi giá trị trong bộ n_estimators với từng giá trị trong max_features). Bộ nào cho ra kết quả tốt nhất sẽ được lựa chọn làm tham số cho mô hình. Sử dụng kỹ thuật nếu bộ dữ liệu tương đối lớn hoặc khi chúng ta có nhiều tham số cần điều chỉnh.*

Random Search

Random search là một kỹ thuật tìm kiếm siêu tham số trong tối ưu hóa mô hình học máy. Nó khác với Grid search trong việc thử tất cả các giá trị tham số trong một lưới các giá trị được định trước. Thay vào đó, Random search sẽ lựa chọn ngẫu nhiên các giá trị trong khoảng giá trị được định trước cho mỗi tham số. Điều này giúp giảm số lượng mô hình phải đào tạo và kiểm tra so với Grid search. Sử dụng kỹ thuật nếu bộ dữ liệu nhỏ hoặc khi chúng ta có ít tham số cần điều chỉnh.

Chương 3

Chương trình minh hoạ

Trong chương này, em sẽ sử dụng các kết quả từ chương trình chạy với một bộ dữ liệu thực tế để minh họa cho một số lý thuyết đã được trình bày ở các chương trước.

3.1 Chương trình minh hoạ bài toán phân lớp nhị phân

3.1.1 Bộ dữ liệu

Bộ dữ liệu này lấy từ một cuộc thi về việc dự đoán một khách hàng có thay đổi nhà mạng viễn thông hay không. Bộ dữ liệu gồm 4250 quan sát được chia thành 2 tập huấn luyện và kiểm thử theo tỉ lệ 7-3. Bộ dữ liệu này gồm 19 trường thuộc tính và 1 trường nhãn. Chi tiết:

- **state** : Mã viết tắt của tiểu bang thuộc Hoa Kỳ.
- **account_length**: Số tháng mà một khách hàng đã sử dụng dịch vụ của nhà cung cấp viễn thông.
- **area_code**: Mã vùng.
- **international_plan**: Khách hàng có sử dụng dịch vụ quốc tế hay không.

- **voice_mail_plan**: Khách hàng có sử dụng dịch vụ gửi/nhận tin nhắn giọng nói không.
- **number_vmail_messages**: Số lượng tin nhắn giọng nói.
- **total_day_minutes**, **total_day_calls**, **total_day_charge**: Lần lượt là tổng số phút gọi, tổng số cuộc gọi và tổng số phút gọi tính phí trong ngày.
- **total_eve_minutes**, **total_eve_calls**, **total_eve_charge**: Lần lượt là tổng số phút gọi, tổng số cuộc gọi và tổng số phút gọi tính phí trong buổi tối.
- **total_night_minutes**, **total_night_calls**, **total_night_charge**: Lần lượt là tổng số phút gọi, tổng số cuộc gọi và tổng số phút gọi tính phí ban đêm.
- **total_intl_minutes**, **total_intl_calls**, **total_intl_charge**: Lần lượt là tổng số phút gọi, tổng số cuộc gọi và tổng số phút gọi tính phí của các cuộc gọi quốc tế.
- **number_customer_service_calls**: Số lần mà khách hàng gọi đến dịch vụ hỗ trợ của nhà cung cấp viễn thông.
- **churn**: Khách hàng có rời mạng hay không. Đây là nhãn của bài toán.

3.1.2 Tiêu chí đánh giá

Ta sẽ sử dụng các tiêu chí đánh giá cho bài toán phân lớp được nêu trong chương 2 cùng với các chỉ số độ lệch và phương sai.

3.1.3 Kết quả chạy chương trình

Sau khi huấn luyện mô hình lần lượt với thuật toán Desicion Tree và Random Forest (*n_estimators*=100) và thực hiên kiểm thử . Ta được kết quả của các tiêu chí đánh giá như sau:

	precision	recall	f1-score	support
0	0.92	1	0.96	1120
1	0.94	0.44	0.60	173
accuracy			0.92	1275
macro avg	0.93	0.72	0.78	1275
weighted avg	0.92	0.92	0.91	1275

Bảng 3.1: Kết quả đánh giá mô hình Decision Tree

	precision	recall	f1-score	support
0	0.93	1	0.96	1120
1	0.95	0.53	0.68	173
accuracy			0.93	1275
macro avg	0.94	0.76	0.82	1275
weighted avg	0.93	0.93	0.92	1275

Bảng 3.2: Kết quả đánh giá mô hình Random Forest

Từ hai bảng kết quả trên ta thấy, các chỉ số như Accuracy, Precision, Recall hay F1-score khi sử dụng mô hình Random forest đều cao hơn khi dùng mô hình Desicion Tree. Vì vậy trong bày toán này thì mô hình Random Forest tỏ ra hiệu quả hơn.

Ta cũng có bảng đánh giá độ lệch và phương sai khi chay 2 mô hình trên:

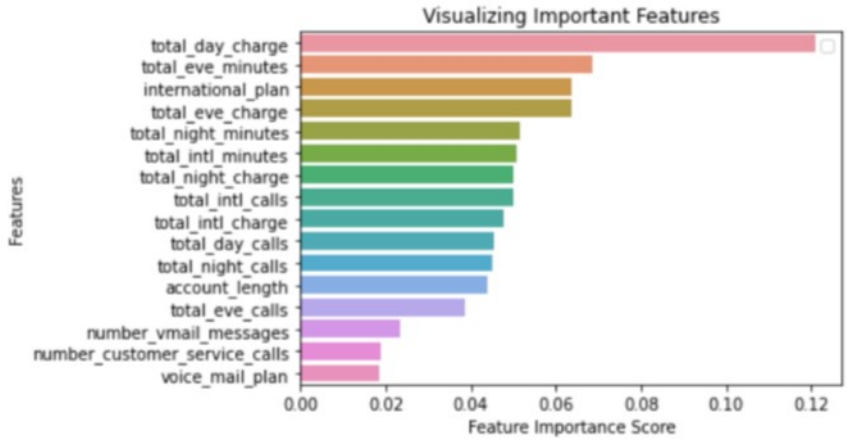
	bias	variance
Decision Tree	0.071	0.094
Random Forest	0.070	0.011

Bảng 3.3: Độ lệch và phương sai của 2 mô hình

Nhận xét: mô hình Random Forest gần như không làm thay đổi độ lệch và giúp

giảm phương sai so với khi chạy mô hình Desicion Tree đơn lẻ.

Dưới đây là xếp hạng độ quan trọng của thuộc tính:



Hình 3.1: Độ quan trọng của thuộc tính

Từ đồ thị trên ta nhận thấy, tổng số phút gọi tính phí trong ngày sẽ ảnh hưởng lớn nhất tới việc khách hàng có rời mạng hay không. Nếu thuộc tính này có giá trị quá thấp nghĩa là đã lâu khách hàng không gọi điện hoặc sim mà khách hàng đang dùng là sim rác dẫn đến khả năng rời mạng cao của khách hàng.

3.2 Chương trình minh hoạ bài toán phân lớp nhiều lớp

3.2.1 Bộ dữ liệu

Xét bài toán dự đoán chất lượng rượu với bộ dữ liệu widequality-red gồm gần 1600 quan sát cùng với 12 thuộc tính. Chất lượng rượu được gán nhãn từ 3 đến 8, tuy nhiên trong phạm vi đồ án, em chỉ huấn luyện và đánh giá mô hình trên các quan sát được gán nhãn là 5, 6, 7. Bài toán trở thành bài toán phân lớp 3 lớp. Chi tiết về bộ dữ liệu như sau:

- **fixed acidity**: Nồng độ axit tartaric

- **volatile acidity**: Tính axit
- **citric acid**: Nồng độ axit Citric
- **residual sugar**: Nồng độ đường dư
- **chlorides**: Nồng độ clo
- **free sulfur dioxide**: Nồng độ acid sulfurus tự do
- **total sulfur dioxide**: Nồng độ acid sulfurus
- **density**: Mật độ (khối lượng/đơn vị thể tích)
- **pH**: Độ pH
- **sulphates**: Nồng độ sunfat
- **alcohol**: Nồng độ chất alcohol
- **quality**: Chất lượng của rượu. Đây là nhãn của bài toán.

3.2.2 Tiêu chí đánh giá

Ta sẽ sử dụng các tiêu chí đánh giá cho bài toán phân lớp được nêu trong chương 2 cùng với các chỉ số độ lệch và phương sai.

3.2.3 Kết quả chạy chương trình

Sau khi huấn luyện mô hình lần lượt với thuật toán Desicion Tree và Random Forest (*n_estimators*=50) và thực hiên kiểm thử, ta được kết quả của các tiêu chí đánh giá như sau:

	precision	recall	f1-score	support
5	0.78	0.75	0.77	154
6	0.60	0.55	0.58	118
7	0.38	0.56	0.46	32
accuracy			0.65	304
macro avg	0.59	0.62	0.60	304
weighted avg	0.67	0.65	0.66	304

Bảng 3.4: Kết quả đánh giá mô hình Decision Tree

	precision	recall	f1-score	support
5	0.82	0.82	0.82	154
6	0.69	0.74	0.71	118
7	0.78	0.56	0.65	32
accuracy			0.76	304
macro avg	0.76	0.71	0.73	304
weighted avg	0.77	0.76	0.76	304

Bảng 3.5: Kết quả đánh giá mô hình Random Forest

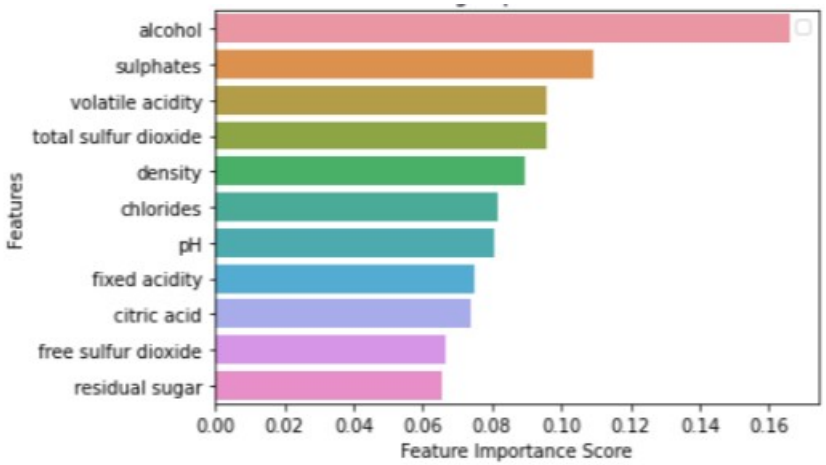
Ta cũng có bảng đánh giá độ lệch và phương sai khi chay 2 mô hình trên:

	bias	variance
Decision Tree	0.240	0.301
Random Forest	0.247	0.140

Bảng 3.6: Độ lệch và phương sai của 2 mô hình

Nhận xét: mô hình Random Forest gần như không làm thay đổi độ lệch và giúp giảm phương sai so với khi chạy mô hình Desicion Tree đơn lẻ.

Dưới đây là xếp hạng độ quan trọng của thuộc tính:



Hình 3.2: Độ quan trọng của thuộc tính

Qua đó ta thấy, nồng độ alcohol sẽ ảnh hướng lớn nhất tới chất lượng rượu, tiếp theo là nồng độ axit tartaric và nồng độ sunfat cũng có độ ảnh hưởng cao đến việc phân loại rượu.

Chương 4

Kết luận

Đồ án đã trình bày những kiến thức cơ bản nhất về bài toán phân lớp và phương pháp Bagging cũng như giới thiệu về 2 thuật toán được sử dụng rất phổ biến trong các bài toán học máy là Decision Tree và Random Forest. Qua đó ta có thể thấy được những ưu điểm của phương pháp Bagging như: giúp giảm phương sai, tăng tính ổn định của bài toán... Tuy nhiên phương pháp này vẫn còn một số hạn chế, tiêu biểu nhất đó là thời gian chạy khá lâu do nó kết hợp nhiều mô hình con. Đồ án cũng đã ứng dụng phương pháp Bagging cho một số bài toán phân lớp với bộ dữ liệu thực tế để đưa ra một cái nhìn tổng quan về phương pháp này.

Tài liệu tham khảo