HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

NGUYỄN MINH HÀ

NGHIÊN CỨU PHÂN LỚP TRÊN DỮ LIỆU MẤT CÂN BẰNG VÀ ỨNG DỤNG

Chuyên ngành: KHOA HỌC MÁY TÍNH

Mã số: 8.48.01.01

TÓM TẮT LUẬN VĂN THẠC SĨ

HÀ NỘI - 2020

Luận văn được hoàn thành tại:

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Người hướng dẫn khoa học: Tiến sĩ VŨ VĂN THỎA

Phản biện 1: ……

Phản biện 2: ……

Luận văn sẽ được bảo vệ trước Hội đồng chấm luận văn thạc sĩ tại Học viện Công nghệ Bưu chính Viễn thông

Vào lúc: ..... giờ ..... ngày ..... tháng .... năm .....

Có thể tìm hiểu luận văn tại:

- Thư viện của Học viện Công nghệ Bưu chính Viễn thông.

PHẦN MỞ ĐẦU

Trong những năm gần đây, vấn đề học máy từ dữ liệu phân bố không cân bằng là một thách thức lớn cho các nhà nghiên cứu trong rất nhiều miền ứng dụng thực tế: mạng internet, bảo mật, viễn thông, quản lý tài chính và tin sinh học… Việc phân tích và hiểu được dữ liệu thô là mục đích của các hệ thống xử lý hỗ trợ ra quyết định ngày càng đóng vai trò quan trọng và trở nên cần thiết. Chúng được áp dụng và đã đạt được nhiều thành công to lớn trong nhiều ứng dụng của cuộc sống như khai phá tri thức, kỹ thuật xử lý dữ liệu, và nhiều ứng dụng khác.

Tuy nhiên, những năm gần đây với sự xuất hiện của dữ liệu phân bố mất cân bằng đang trở thành nguyên nhân gây ra nhiều khó khăn ảnh hưởng đến các thuật toán học máy chuẩn, những thuật toán được thiết kế và áp dụng vào ứng dụng của dữ liệu phân bố cân bằng. Khi những thuật toán chuẩn này được áp dụng vào dữ liệu mất cân bằng, chúng xử lý dữ liệu lệch lạc, dẫn đến không đạt được độ chính xác cao giữa các lớp của dữ liệu.

Thêm vào đó, vấn đề phân bố dữ liệu mất cân bằng đang ngày càng trở nên quan trọng trong thực tế, với lượng lớn các ứng dụng. Khi áp dụng các thuật toán phân lớp truyền thống lên các tập dữ liệu mất cân bằng, đa số các phần tử thuộc lớp đa số sẽ được phân lớp đúng và các phần tử thuộc lớp thiểu số cũng sẽ được gán nhãn lớp là nhãn lớp của lớp đa số. Điều này dẫn đến kết quả là độ chính xác (accuracy) của việc phân lớp có thể rất cao, trong khi giá trị độ nhạy (sensitivity) lại rất thấp.

Xuất phát từ thực tế và mục tiêu như trên, học viên chọn thực hiện đề tài luận văn tốt nghiệp chương trình đào tạo thạc sĩ có tên "**Nghiên cứu phân lớp trên dữ liệu mất cân bằng và ứng dụng**".

Nội dung của luận văn ngoài phần mở đầu, kết luận gồm các chương chính như sau.

Chương 1: *Khảo sát tổng quan về phân lớp dữ liệu, học máy và các vấn đề liên quan.*

Chương 2: *Chương này nghiên cứu một số thuật toán để giải quyết bài toán phân lớp dữ liệu mất cân bằng.*

Chương 3: *Thử nghiệm phân lớp dữ liệu mất cân bằng dựa trên các thuật toán đã nghiên cứu trong chương 2.*

Phần kết luận tóm tắt lại các nội dung đã đạt được của luận văn, và nêu lên một số gợi ý về hướng phát triển tiếp theo của luận văn.

CHƯƠNG 1. TỔNG QUAN VỀ BÀI TOÁN PHÂN LỚP DỮ LIỆU TRÊN CÁC DỮ LIỆU MẤT CÂN BẰNG

1.1. Giới thiệu về bài toán phân lớp dữ liệu

1.1.1. Khái niệm về phân lớp dữ liệu và bài toán phân lớp dữ liệu

- *Phân lớp dữ liệu:*

Phân lớp dữ liệu(classification) là một trong những hướng nghiên cứu chính của khai phá dữ liệu. Thực tế đặt ra nhu cầu là từ một cơ sơ dữ liệu với nhiều thông tin ẩn con người có thể rút trích ra các quyết định nghiệp vụ thông minh. Phân lớp là một dạng của phân tích dữ liệu nhằm rút trích ra một mô hình mô tả các lớp dữ liệu quan trọng hay dự đoán xu hướng dữ liệu trong tương lai.

Phân lớp dự đoán giá trị của những nhãn xác định hay những giá trị rời rạc, có nghĩa là thao tác với những đối tượng dữ liệu mà có bộ giá trị là biết trước. Cụ thể, phân lớp là quá trình nhóm các đối tượng giống nhau vào một lớp dựa trên các đặc trưng dữ liệu của chúng.

- *Bài toán phân lớp dữ liệu:*

Là quá trình phân lớp một đối tượng dữ liệu vào một hay nhiều lớp đã cho trước nhờ một mô hình phân lớp (model). Mô hình này được xây dựng dựa trên một tập dữ liệu được xây dựng trước đó có gán nhãn (còn gọi là tập huấn luyện). Quá trình phân lớp là quá trình gán nhãn cho đối tượng dữ liệu.

Bài toán phân lớp dữ liệu có thể phát biểu tổng quát như sau:

Cho $U = \{A_1, A_2, ..., A_m\}$ là tập có m thuộc tính, $Y = \{y_1, y_2, ..., y_n\}$ là tập các nhãn của lớp: với $D = A_1 \times \ldots \times A_m$ là tích Đề - các của các miền của m thuộc tính tương ứng có n số lớp và N là số mẫu dữ iệu. Mỗi dữ liệu $d_i \in D$ thuộc một lớp $y_i \in Y$ tương ứng tạo thành từng cặp $(d_i, y_i) \in (D, Y)$.

1.1.2. Quy trình thực hiện phân lớp dữ liệu:

Quy trình thực hiện phân lớp dữ liệu thường được thực hiện theo 2 bước: Bước thứ nhất (learning) quá trình học và bước thứ hai phân lớp dữ liệu mới.

- *Bước thứ nhất (learning)*

Đầu vào của quá trình này là một tập dữ liệu có cấu trúc được mô tả bằng các thuộc tính và được tạo ra từ tập các bộ giá trị của các thuộc tính đó. Mỗi bộ giá trị được gọi chung là một phần tử dữ liệu (data tuple), có thể là các mẫu (sample), ví dụ (example)… Trong tập dữ liệu này, mỗi phần tử dữ liệu được giả sử thuộc về một lớp định trước, lớp ở đây là giá trị

của một thuộc tính được chọn làm thuộc tính gán nhãn lớp hay thuộc tính phân lớp (class lable attribute). Đầu ra của bước này thường là các quy tắc phân lớp dưới dạng luật dạng if-then, cây quyết định, công thức logic, hay mạng nơron.

- *Bước thứ hai (classification)*

Bước thứ hai dùng mô hình đã xây dựng ở bước thứ nhất để phân lớp dữ liệu mới. Holdout là một kỹ thuật đơn giản để ước lượng độ chính xác đó. Kỹ thuật này sử dụng một tập dữ liệu kiểm tra với các mẫu đã được gán nhãn lớp. Các mẫu này được chọn ngẫu nhiên và độc lập với các mẫu trong tập dữ liệu đào tạo. Độ chính xác của mô hình trên tập dữ liệu kiểm tra đã đưa là tỉ lệ phần trăm các mẫu trong tập dữ liệu kiểm tra được mô hình phân lớp đúng (so với thực tế). Nếu độ chính xác của mô hình được ước lượng dựa trên tập dữ liệu đào tạo thì kết quả thu được là rất khả quan vì mô hình luôn có xu hướng "quá vừa" dữ liệu. Do vậy cần sử dụng một tập dữ liệu mà giá trị của thuộc tính phân lớp là chưa biết.

1.1.3. Các độ đo đánh giá mô hình phân lớp dữ liệu

Quá trình đánh giá mô hình phân lớp thường chia làm 2 phần hay hướng tiếp cận: *phân chia bộ dữ liệu để huấn luyện* và *kiểm chứng mô hình*.

Một số tiêu chí mô tả độ hiệu quả của mô hình phân lớp:

- Accuracy: khả năng mô hình phân lớp dự báo, phân loại hay xác định đúng class cho dữ liệu cần phân loại.
- Speed: tốc độ hay khả năng mô hình đưa ra kết quả phân tích nhanh chóng, nó còn liên qua đến chi phí tính toán khi xây dựng, và sử dụng mô hình.
- Robustness: khả năng của mô hình xử lý nhiễu hoặc dữ liệu với các giá trị bị thiếu và đưa ra dự đoán chính xác.
- Scalability: Phương pháp hay khả năng xây dựng mô hình phân lớp hiệu quả trong xử lý, phân tích lượng lớn dữ liệu.
- Interpreability: là khả năng giải thích, mứa độ phức tạp của mô hình hay nói cách khác cấu trúc mô hình, phương pháp xây dựng mô hình có dễ hiểu hay không.

Có 2 phương pháp đánh giá phổ biến là holdout và cross-validation.

- ***Holdout:***

Holdout, là phương pháp phân chia ngẫu nhiên tập dữ liệu thành 2 tập dữ liệu độc lập là: tập dữ liệu huấn luyện và tập kiểm định mô hình. Cụ thể trong phương pháp Holdout ta sẽ có các tập dữ liệu:

- Training set: dữ liệu phục vụ xây dựng mô hình, xác định các thuật toán, biến dữ liệu

phù hợp

- Validation set: là dữ liệu được sử dụng để đánh giá hiệu suất của mô hình được xây dựng trong giai đoạn huấn luyện, hỗ trợ thử nghiệm để tinh chỉnh các tham số mô hình và chọn mô hình hoạt động tốt nhất.
- Test set: là dữ liệu được sử dụng để đánh giá độ hiệu quả của mô hình, mức độ chính xác trong việc phân loại dữ liệu (không chứa nhãn phân loại).

Thường tỉ lệ phân chia cho training data set là 70% và test data set là 30%. Ưu điểm của Holdout là nhanh chóng, đơn giản và linh hoạt.

- ***Cross - validation:***

Cross - validation là một kỹ thuật phân chia tập dữ liệu ban đầu thành training data được sử dụng để huấn luyện mô hình và một tập dữ liệu độc lập được sử dụng để đánh giá. Phương pháp này lặp lại nhiều lần cho đến khi có k số mô hình khác nhau, sao cho mỗi lần, một trong các tập k được sử dụng làm tập kiểm thử các tập còn lại khác được ghép lại với nhau tạo thành tập huấn luyện. Việc ước tính độ chính xác hay lỗi (accuracy hay error) được tính trung bình trên tất cả các thử nghiệm k để đánh giá mức độ hiệu quả của cả mô hình.

- ***Confusion Matrix***

Là một phương pháp đánh giá kết quả của những bài toán phân loại với việc xem xét cả những chỉ số về độ chính xác và độ bao quát của các dự đoán cho từng lớp. Một confusion matrix gồm 4 chỉ số sau với mỗi lớp phân loại:

Sử dụng bài toán về chẩn đoán ung thư để giải thích 4 chỉ số này. Trong bài toán chẩn đoán ung thư ta có 2 lớp: lớp bị ung thư được chẩn đoán Positive và lớp không bị ung thư được chẩn đoán là Negative:

- TP (True Positive): Số lượng dự đoán chính xác. Là khi mô hình dự đoán đúng một người bị ung thư.
- TN (True Negative): Số lượng dự đoán chính xác một cách gián tiếp. Là khi mô hình dự đoán đúng một người không bị ung thư, tức là việc không chọn trường hợp bị ung thư là chính xác.
- FP (False Positive - type 1 error): Số lượng các dự đoán sai lệch. Là khi mô hình dự đoán một người bị ung thư và người đó hoàn toàn khỏe mạnh.
- FN (False Negative - type 2 error): Số lượng các dự đoán sai lệch một cách gián tiếp. Là khi mô hình dự đoán một người không bị ung thư nhưng người đó bị ung thư, tức là việc không chọn trường hợp bị ung thư là sai.

Từ 4 chỉ số này, ta có 2 đại lượng để đánh giá mức độ tin cậy của một mô hình:

- Precision: trong tất các các dự đoán Positive được đưa ra, bao nhiêu dự đoán là chính xác? Chỉ số này được tính theo công thức:

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

- Recall: Trong tất cả các trường hợp Positive, bao nhiêu trường hợp đã được dự đoán chính xác? Chỉ số này được tính theo công thức:

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

Giả sử có 1 tập dữ liệu gồm 100 người với 90 người khỏe mạnh (Negative) và 10 người mắc bệnh ung thư (Positive) và mô hình dự đoán đúng 2/10 người bị ung thư, tức là đưa ra dự đoán 2 người bị ung thư thì cả 2 dự đoán đều chính xác. Như vậy, chỉ số Precision khi dự đoán lớp ung thư là 1.

Tuy nhiên, 8/10 người còn lại đã bị bỏ qua, từ đó chỉ số về Recall chỉ là 0.2 - con số rất thấp. Để đánh giá độ tin cậy chung của mô hình, người ta đã kết hợp 2 chỉ số Precision và Recall thành 1 chỉ số duy nhất: F - score, được tính theo công thức:

$$\text{F - measure} = \frac{2 * \text{Re}\,call\ * \text{Pr}\,ecision}{\text{Re}\,call\ + \text{Pr}\,ecision}$$

1.2. Dữ liệu mất cân bằng

1.2.1. Khái niệm về dữ liệu mất cân bằng

Dữ liệu thu thập được trong thực tế xuất hiện nhiều các bộ dữ liệu mất cân bằng, nghĩa là trong tập dữ liệu có sự chênh lệch lớn về số lượng các phần tử giữa các lớp. Lớp có nhiều phần tử hơn ta gọi là lớp đa số, lớp có ít phần tử hơn ta gọi là lớp thiểu số. Các bộ dữ liệu trong nhiều ứng dụng thực tế, chẳng hạn như phát hiện các giao dịch gian lận, phát hiện xâm nhập mạng, dự đoán rủi ro trong quản lý, chẩn đoán y khoa, …

1.2.2. Các đặc điểm phân lớp dữ liệu mất cân bằng:

Sự chênh lệch về số lượng giữa lớp đa số và lớp thiểu số làm cho việc phân lớp đúng các mẫu thuộc lớp thiểu số bị giảm hiệu quả. Mức độ mất cân bằng của bộ dữ liệu được biểu thị bằng tỷ lệ giữa số lượng mẫu của hai lớp. Tỷ lệ mất cân bằng của tập dữ liệu càng cao thì việc phát hiện đúng các mẫu của lớp thiểu số càng khó khăn. Trong các ứng dụng thực tế, tỷ lệ mất cân bằng có thể là 1:100, 1:1000, … thậm chí có thể hơn.

1.2.3. Các ứng dụng của phân lớp dữ liệu mất cân bằng

Bài toán phân lớp dữ liệu có rất nhiều ứng dụng trong các lĩnh vực khoa học, công nghệ và đời sống xã hội như: Trong ngành y tế, phân tích thị trường bán lẻ, trong ngành giáo dục, Quy trình sản xuất, Phát hiện gian lận, Hỗ trợ điều tra tội phạm, Ngành tài chính - ngân hàng…..

1.3. Tổng quan kỹ thuật xử lý dữ liệu mất cân bằng

Có thể phân chia các kỹ thuật thành hai hướng tiếp cận chính: hướng tiếp cận ở mức độ dữ liệu và hướng tiếp cận ở mức độ thuật toán.

1.3.1. Hướng tiếp cận ở mức độ dữ liệu

Tiếp cận ở mức độ dữ liệu có mục tiêu điều chỉnh tỉ lệ mất cân bằng giữa hai lớp trong bộ dữ liệu. Các phương pháp ở hướng tiếp cận này có nhiều hình thức khác nhau của việc lấy mẫu như: sinh thêm các phần tử lớp thiểu số (sinh ngẫu nhiên, sinh thêm phần tử nhân tạo, …), loại bỏ các phần tử lớp đa số, hoặc kết hợp cả hai phương pháp trên.

- *Sinh thêm phần tử lớp thiểu số*

Hiện nay, có nhiều phương pháp sinh thêm phần tử cho lớp lớp thiểu số như: sinh ngẫu nhiên phần tử lớp thiểu số, lựa chọn phần tử lớp thiểu số, hay sinh thêm mẫu nhân tạo.

Sinh ngẫu nhiên các phần tử ở lớp thiểu số (Random Over-sampling) là phương pháp đơn giản nhất nhằm cân bằng phân lớp thông qua việc nhân bản ngẫu nhiên các mẫu lớp thiểu số. Ý tưởng của phương pháp này là lựa chọn ngẫu nhiên các mẫu thuộc lớp thiểu số và nhân bản chúng tạo ra mẫu mới giống hệt chúng.

- *Loại bỏ phần tử lớp đa số*

Loại bỏ phần tử lớp đa số là phương pháp điều chỉnh phân bố dữ liệu bằng cách giảm bớt số lượng phần tử lớp đa số.

Loại bỏ một cách ngẫu nhiên các mẫu thuộc lớp đa số (Random under -sampling) là cách đơn giản nhất. Phương pháp này thực hiện loại bỏ ngẫu nhiên phần tử thuộc lớp đa số trong tập huấn luyện cho tới khi có được tỷ lệ phù hợp giữa hai lớp. Vì lý do này, số lượng phần tử trong tập huấn luyện giảm đáng kể.

1.3.2. Hướng tiếp cận ở mức độ thuật toán

Tiếp cận ở mức độ thuật toán nghĩa là điều chỉnh các thuật toán phân lớp để tăng độ chính xác khi phân lớp đối với dữ liệu mất cân bằng. Chiến lược chung để đối phó với vấn

đề mất cân bằng trong các bộ dữ liệu là lựa chọn một khuynh hướng quy nạp thích hợp. Các kỹ thuật sau đây có thể giúp đào tạo một bộ phân loại để phát hiện ra lớp bất thường.

1.3.2.1. Sử dụng các chỉ số đánh giá phù hợp

Các chỉ số đánh giá thay thế khác có thể được áp dụng như:

- Độ chính xác / độ đặc hiệu: bao nhiêu trường hợp được chọn có liên quan. Nhớ lại / Độ nhạy: bao nhiêu trường hợp có liên quan được chọn.
- Điểm số F1: trung bình hài hòa của độ chính xác và thu hồi.
- MCC: hệ số tương quan giữa phân loại nhị phân được quan sát và được dự đoán.
- AUC: mối quan hệ giữa tỷ lệ thực dương và tỷ lệ dương tính giả.

1.3.2.2. Sử dụng K - fold Cross - Validation đúng cách

Đáng chú ý là cross - validation phải được áp dụng đúng cách trong khi sử dụng phương pháp over - sampling để giải quyết các vấn đề mất cân đối.

1.3.2.3. Tập hợp các tập dữ liệu được lấy mẫu khác nhau

Cách dễ nhất để khái quát hóa mô hình thành công là sử dụng nhiều dữ liệu hơn. Vấn đề là các bộ phân loại out - of - the - box như hồi quy logistic hoặc rừng ngẫu nhiên có xu hướng tổng quát hóa bằng cách loại bỏ lớp hiếm. Một thực hiện tốt nhất là xây dựng n mô hình sử dụng tất cả các mẫu của các mẫu hiếm và n - khác biệt của lớp phong phú.

1.3.2.4. Lấy mẫu với các tỷ lệ khác nhau

Cách tiếp cận trước đó có thể được tinh chỉnh bằng cách thay đổi tỷ lệ giữa lớp hiếm và phong phú. Tỷ lệ tốt nhất phụ thuộc nhiều vào dữ liệu và các mô hình được sử dụng. Nhưng thay vì đào tạo tất cả các mô hình với tỷ lệ tương tự nhau, có thể tổng hợp các tỷ lệ khác nhau. Vì vậy, nếu 10 mô hình được đào tạo, có thể điều chỉnh để một mô hình có tỷ lệ 1:1 (hiếm: phong phú) và một mô hình khác với 1:3, hoặc thậm chí 2:1.

1.4. Kết luận chương 1

Chương I của luận văn đã giới thiệu về bài toán phân lớp dữ liệu và quy trình phân lớp dữ liệu, các độ đo đánh giá các mô hình phân lớp dữ liệu và một số ứng dụng.

Chương này luận văn cũng trình bày về dữ liệu mất cân bằng, các đặc điểm của phân lớp dữ liệu mất cân bằng cũng như một số kỹ thuật xử lý dữ liệu mất cân bằng.

CHƯƠNG 2. MỘT SỐ THUẬT TOÁN PHÂN LỚP DỮ LIỆU MẤT CÂN BẰNG

2.1. Thuật toán DEC - SVM

2.1.1. Giới thiệu thuật toán

Thuật toán máy vector hỗ trợ (Support Vector Machines - SVM) thường được sử dụng xây dựng các bộ phân lớp dữ liệu.

Định lý 2.1 sau đây đảm bảo cơ sở toán học cho SVM [7].

Định lý 2.1: Cho tập hợp gồm m điểm trong không gian R^d. Ta chọn một điểm nào đó trong chúng làm điểm gốc và tạo thành m-1 vector điểm. Khi đó m điểm đã cho có thể được phân tách bởi một siêu phẳng có hướng khi và chỉ khi tập hợp các vector điểm là độc lập tuyến tính.

Khoảng cách của điểm dữ liệu gần nhất của mỗi lớp đến siêu phẳng phân tách gọi là biên (hay lề). Trong số các siêu phẳng thỏa mãn định lý 2.1, siêu phẳng tối ưu có biên lớn nhất sẽ được lựa để phân tách các điểm. Các kỹ thuật SVM nhằm nghiên cứu xây dựng các siêu phẳng tối ưu này một cách hiệu quả nhất.

Ưu điểm nổi bật của phương pháp SVM là thực hiện tối ưu toàn cục cho mô hình phân lớp. Do đó, mô hình SVM có chất lượng cao, chịu đựng được nhiễu. Mặt khác, SVM là một phương pháp tốt (phù hợp) đối với những bài toán phân lớp có không gian biểu diễn thuộc tính lớn. Các đối tượng cần phân lớp được biểu diễn bởi một tập rất lớn các thuộc tính.

Tuy nhiên khi áp dụng trực tiếp thuật toán SVM cho phân lớp dữ liệu mất cân bằng có thể không đạt được kết quả mong muốn. Lý do là SVM thường có xu hướng thiên vị đối với lớp đa số và bỏ qua lớp thiểu số (xử lý chúng như là nhiễu) [8]. Việc phân loại sai các mẫu thuộc lớp thiểu số có thể gây nên những tổn thất lớn đối với các bài toán thực tế.

Để khắc phục vấn đề trên, phương pháp sinh thêm phần tử nhân tạo cho lớp thiểu số là phương pháp khá phổ biến thường được sử dụng. Tuy nhiên, trong nhiều trường hợp, việc sinh thêm mẫu có thể sẽ tạo ra những mẫu dư thừa hoặc nhiễu làm ảnh hưởng tới hiệu quả phân lớp.

Do đó, trước khi sử dụng SVM cần áp dụng thuật toán DEC (a novel Differential Evolution Clustering hybrid resampling- DEC) [9] để điều chỉnh dữ liệu cho bài toán phân lớp dữ liệu mất cân bằng. Thuật toán DEC là sự kết hợp giữa phương pháp sinh thêm phần

tử cho lớp thiểu số và sử dụng kỹ thuật phân cụm K-means để loại bỏ bớt phần tử dư thừa, nhiễu trong dữ liệu. Với mỗi mẫu thuộc lớp thiểu số, tạo ra một mẫu đột biến từ hai trong số những láng giềng gần nó nhất, sau đó sử dụng thuật toán di truyền để sinh thêm phần tử cho lớp thiểu số từ mẫu thiểu số ban đầu và mẫu đột biến mới tạo ra. Sau khi điều chỉnh dữ liệu bằng thuật toán DEC, thuật toán SVM sẽ được sử dụng để xây dựng mô hình phân lớp.

Như vậy, thuật toán DEC-SVM có thể xem như sự kết hợp giữa hai thuật toán DEC và SVM. Nội dung trình bày trong mục này tham khảo từ tài liệu [4].

2.1.2. Khảo sát nội dung thuật toán

2.1.2.1. Điều chỉnh dữ liệu bằng thuật toán DE (Differential Evolution over-sampling)

Với thuật toán SMOTE, mẫu mới sẽ được sinh ra từ một mẫu positive ban đầu và một trong những láng giềng của nó. Với nền tảng là thuật toán SMOTE, tuy nhiên, trong thuật toán DE, từ hai trong số các láng giềng gần nhất của một mẫu positve sẽ tạo ra một mẫu "đột biến", và mẫu mới được sinh ra bằng cách lai ghép chéo mẫu đột biến này và mẫu positive ban đầu.

- *Đột biến*

Trong tập dữ liệu huấn luyện, đầu tiên chọn ngẫu nhiên một mẫu positive x_i và tìm k láng giềng gần nhất của nó, sau đó chọn ngẫu nhiên hai láng giềng trong k láng giềng đó: x_{n1} và x_{n2} . Một mẫu đột biến x_{mu} sẽ được tạo ra bằng cách sử dụng công thức (1) với *rand(0,1)* là hằng số ngẫu nhiên trong khoảng [0,1]:

$$x_{mu} = x_i + rand(0,1) \times (x_{n1} - x_{n2}) \quad (1)$$

- *Crossover:*

Qua bước đột biến, ta tạo ra số lượng mẫu đột biến đúng bằng số lượng mẫu positive ban đầu trong tập dữ liệu huấn luyện. Ở bước này, ta sẽ sử dụng các mẫu đột biến cùng với các mẫu positive ban đầu để tạo ra mẫu nhân tạo mới.

Cụ thể, các mẫu mới sẽ được hình thành dựa theo (2):

$$x_{new,j} = \begin{cases} x_{i,j} & if\ (rand(j) > CR)\ and\ j \neq rand(s) \\ x_{mu,j} & if\ (rand(j) < CR)\ or\ j = rand(s) \end{cases} \quad (2)$$

Trong đó $x_{i,j}$ đại diện cho thuộc tính thứ j của mẫu thứ i

CR là hằng số crossover được lựa chọn ngẫu nhiên trong [0, 1] và được xác định trước bởi người dùng.

rand(j) là giá trị được lựa chọn ngẫu nhiên trong khoảng [0, 1].

Giá trị của biến *rand(s)* là chỉ số của các thuộc tính được lấy một cách ngẫu nhiên, đảm bảo rằng mẫu mới sinh ra sẽ có ít nhất một thuộc tính từ mẫu đột biến.

Số mẫu nhân tạo được tạo ra đúng bằng số mẫu nhân tạo ban đầu. Tùy thuộc vào số lượng mẫu possitive cần lấy, lặp lại các bước đột biến và crossover cho dữ liệu huấn luyện.

2.1.2.2. Kỹ thuật làm sạch dữ liệu sử dụng phân cụm

Sau khi thực hiện thuật toán DE, dữ liệu thu được đã được cải thiện hơn về tỉ lệ giữa hai lớp. Tuy nhiên, không loại trừ khả năng sinh ra những mẫu dư thừa hoặc nhiễu. Để khắc phục, ta sẽ sử dụng kỹ thuật phân cụm để phân cụm cho tập dữ liệu với mục đích loại bỏ những mẫu không cần thiết.

Nếu như tất cả các mẫu trong một cụm đều có cùng một nhãn lớp (tức là hoặc cùng là positive hoặc cùng là negative), ta sẽ tiến hành loại bỏ những mẫu dư thừa hoặc nhiễu. Ví vụ với cụm F có chứa tất cả các mẫu negative, ta sẽ thực hiện theo những bước sau:

- Xác định ngưỡng tương đồng trong [0,1]
- Tính $\bar{x}$ theo công thức (3)

$$\bar{x} = \frac{1}{n_i}\sum_{j=1}^{n_i} x_i \qquad (3)$$

- Tìm mẫu trung tâm $\bar{x}_c \in F$ gần $\bar{x}$ nhất
- Tính độ tương đồng S_{ic} giữa mỗi đầu $x_i \in F$ và $\bar{x}_c$ theo (4). Nếu S_{ic} lớn hơn ngưỡng tương đồng thì x_i sẽ bị loại khỏi F

$$S_{ic} = \frac{\sum_{k=1}^{n} x_{ik} * x_{jk}}{\sqrt{(\sum_{k=1}^{n} x^2{}_{ik})(\sum_{k=1}^{n} x^2{}_{jk})}} \qquad (4)$$

Ngưỡng tương đồng càng nhỏ thì càng nhiều mẫu bị loại bỏ

Trong đó: n_i là số lượng mẫu trong cụm thứ i, x_{ik} là thuộc tính thứ k của mẫu x_i, S_{ij} là độ tương đồng giữa x_i và x_j

2.1.2.3. Thuật toán DEC-SVM

Sau khi sử dụng thuật toán DEC để điều chỉnh dữ liệu, ta sử dụng thuật toán SVM để phân lớp cho cho tập dữ liệu huấn luyện tạo nên một mô hình phân lớp.

Nội dung thuật toán DEC-SVM có thể trình bày dưới dạng giả mã của như sau:

DEC-SVM(N, m, K, s, T)

Input: Số mẫu lớp thiểu số N, số thuộc tính m, số cụm K, ngưỡng tương đồng s, số lượng của DE là T%.

Output: Mô hình huấn luyện

```
Void DEC-SVM() {
/******** Sinh thêm mẫu bằng DE ********/
st = 0;
G = int(N*T%); //số mẫu lớp thiểu số được tạo ra
For (t = 0; t< ceil(T/100); t++) {
    For (i=0; i<N; i++) {
     /*Đánh giá khoảng cách giữa tất cả các mẫu thiểu số*/
        For (p=0; p<N; p++) {
            For (j=0; j<m; j++){
            Euclidean_distance[i][j] += pow((x[i][j] - x[p][j]), 2);
            }
        }
        Tìm hai mẫu x(n1) và x(n2) trong k láng giềng gần nhất của x(i); For(j =
        0; j < m; j++) {//Đột biến và crossover của DE
            If(rand_j > CR && j != rand_s) {
            x[N+st][j] = x[i][j];
            }
            Else if(rand_j <=CR || j == rand_s){
            x[N+st][j] = x[i][j] + rand(0,1)(x[n1][j] - x[n2][j]);
            }
        }
        st++;
        If (st==G) break;
    }
    If (st==G) break;
}
/********Làm sạch dữ liệu bằng phân cụm********/
    Phân cụm tập dữ liệu thành K cụm;
```

```
Để lại những cụm có các mẫu mang nhãn hỗn hợp For(t = 0;
    t < K; t++) {
    /* Tính trung bình của các cụm mà tất cả các mẫu có cùng nhãn*/
    If (cụm t có tất cả các mẫu cùng nhãn){
    n = số mẫu trong cụm t;
        For(j = 0; j < m; j++) {
            For(i = 0; i < n; i++) { Mean[j] +=
                x[i][j];
            }
            Mean[j] = Mean[j] / n;
        }
        Tìm mẫu trung tâm x[p] của cụm t;
        For (i = 0; i < n; i++) {//Loại bỏ các mẫu dư thừa
            Tính độ tương đồng s[i][p]; If
            (s[i][p] >s) Loại bỏ x[i];
        }
    }
    }
/********Phân lớp bằng SVM********/
SVM training;
SVM prediction;
}
```

2.1.3. Đánh giá thuật toán

Thuật toán DEC-SVM có thể chia làm hai pha: tiền xử lý dữ liệu và pha phân lớp dữ liệu.

Pha tiền xử lý dữ liệu gồm hai bước:

Bước 1: Điều chỉnh dữ liệu bằng thuật toán DE.

Bước 2: Làm sạch dữ liệu sử dụng phân cụm.

Pha phân lớp dữ liệu sử dụng thuật toán SVM.

Thuật toán DEC-SVM so với thuật toán SVM chỉ bổ sung thêm phần tiền xử lý dữ liệu với độ phức tạp tính toán cộng thêm là $O(N^3*m)$.

2.2. Thuật toán HMU

2.2.1. Giới thiệu thuật toán

Phương pháp sinh phần tử ngẫu nhiên *(Random Oversampling)* là phương pháp sinh thêm phần tử đơn giản nhất bằng cách tăng số lượng một số phần tử được chọn ngẫu nhiên thuộc lớp thiểu số để cân bằng tỷ lệ.

Phương pháp giảm số phần tử ngẫu nhiên *(Random Undersampling)* sẽ chọn ngẫu nhiên và loại bỏ một số phần tử thuộc lớp đa số để làm giảm tỷ lệ mất cân bằng của các tập dữ liệu. Trong mục này luận văn sẽ khảo sát thuật toán HMU (Hypothesis Margin based Undersampling - HMU).Ý tưởng của thuật toán HMU là làm giảm số phần tử thuộc lớp đa số mới nhắm tới xử lý các đối tượng khó phân lớp, khắc phục nhược điểm đã đề cập ở trên.

2.2.2. Khảo sát nội dung thuật toán

2.2.2.1. Phân loại lề

Lề (margin), đóng vai trò quan trọng trong lĩnh vực học máy, thể hiện tính hiệu quả khi phân lớp của bộ phân lớp (classifier).

Có hai cách xác định giá trị lề cho một phần tử dựa trên quy tắc phân lớp [10]. Cách thứ nhất là đo khoảng cách từ phần tử đang xét tới biên quyết định được xác định bởi bộ phân lớp và lề trong trường hợp này gọi là lề phần tử (sample margin). Đối với cách thứ hai, lề là khoảng cách mà bộ phân lớp có thể di chuyển sao cho không làm thay đổi nhãn lớp của các phần tử đã được xác định, và được gọi là lề giả thuyết (hypothesis margin).

2.2.2.2. Thuật toán HMU

Thuật toán HMU được mô tả dưới dạng giả code như sau: **HMU Algorithm**

***Input**: lớp đa số **N**; số lượng phần tử cần loại bỏ **d**; **Output**:*
*lớp đa số sau khi đã làm giảm số phần tử **N*;***

Begin

1. *nos = |N| -d*
2. *N* = N*
3. *While (|N*| > nos)*
4. *Tính giá trị lề mar(x) của tất cả các phần tử x thuộc N* trên toàn bộ tập dữ liệu và lưu vào mảng @margin*
5. *Sắp xếp mảng @margin*
6. *Loại bỏ phần tử có giá trị lề tương ứng bé nhất trong mảng @margin*

7. *Cập nhật lại N**
8. *End while*

End

Ghi chú:

- Kích thước của lớp đa số sau khi làm giảm bớt số phần tử N* được xác định dựa vào số lượng phần tử cần loại bỏ d. Chỉ số d này phụ thuộc vào từng tập dữ liệu cụ thể.

- Khoảng cách được sử dụng để xác định lề là khoảng cách Euclidean.

Sau khi sử dụng HMU có thể sử dụng các thuật toán học máy truyền thống như SVM để xây dựng các bộ phân lớp dữ liệu đối với các dữ liệu mất cân bằng.

Có thể đề xuất thuật toán phân lớp dữ liệu HMU-SVM như sau:

```
Void HMU-SVM() {
        Thực hiện HMU tiền xử lý dữ liêu;
                SVM training;
                SVM prediction;
        }
```

2.2.3. Đánh giá thuật toán

Thuật toán HMU ứng dụng trong pha tiền xử lý dữ liệu trong bài toán phân lớp dữ liệu mất cân bằng. Với lớp đa số gồm N phần tử và số lượng phần tử cần loại bỏ từ lớp đa số là d thì độ phức tạp tính toán là $O(N^2*d)$.

So sánh với thuật toán DEC-SVM thì thuật toán HMU trong pha tiền xử lý dữ liệu có độ phức tạp tính toán nhỏ hơn. Tuy nhiên, do HMU làm giảm lớp đa số nên thường chỉ được áp dụng khi tập dữ liệu cân mất cân bằng có lớp thiểu số đủ lớn để đảm bảo hiệu suất phân lớp. Trong trường hợp lớp thiểu số quá thưa thì HMU sẽ không cho kết quả tốt.

2.3. Thuật toán HBU

2.3.1. Giới thiệu thuật toán

Trong mục này, luận văn khảo sát thuật toán HBU (Hypothesis margin based Borderline Under-sampling - HMU) nhằm khắcc phục những nhược điểm của HMU. Ý tưởng của thuật toán HBU là dựa vào giá trị lề giả thuyết và ưu tiên loại bỏ các phần tử nằm ở biên. Với mỗi phần tử lớp thiểu số x, một số lượng n các phần tử lớp đa số nằm gần x nhất sẽ được chọn để loại bỏ. Giá trị n thay đổi phụ thuộc vào giá trị lề của mỗi x khác nhau.

2.3.2. Khảo sát nội dung thuật toán

Thuật toán HBU có thể mô tả dưới dạng giả code như sau: ***HBU Algorthim***

Input: tập các phần tử lớp thiểu số P; tập các phần tử lớp đa số N; số lượng phần tử cần loại bỏ N' ; tham số k

Output: tập các phần tử lớp đa số mới N;*

1. *Begin*
2. *Tính giá trị lề mar(x) của tất cả phần tử lớp thiểu số x trên tập dữ liệu đã cho*
3. $max = \max_{x \in P} mar(x)$
4. $min = \min_{x \in P} mar(x)$
5. $p = |P|$
6. *Foreach x in P*
7. ***nos***$= int((N'/p)*(k +(max-mar(x))/(max-min)));$
8. *Loại bỏ* ***nos*** *phần tử lớp đa số mà gần với x nhất*
9. $N' = N' - nos$
10. $p = p - 1$
11. *End-for*
12. *End*

Kích thước của lớp đa số sau khi làm giảm bớt số phần tử N* được xác định dựa vào số lượng phần tử cần loại bỏ N', giá trị này phụ thuộc vào từng tập dữ liệu cụ thể.

Tương tự HMU, trong pha tiền xử lý dữ liệu có thể sử dung HBU và sau đó sử dụng các thuật toán học máy truyền thống như SVM để xây dựng các bộ phân lớp dữ liệu đối với các dữ liệu mất cân bằng.

Có thể đề xuất thuật toán phân lớp dữ liệu HBU-SVM như sau:

```
Void HBU-SVM() {
        Thực hiện HBU tiền xử lý dữ liêu;
                SVM training;
                SVM prediction;
        }
```

2.3.3. Đánh giá thuật toán

Thuật toán HBU ứng dụng trong pha tiền xử lý dữ liệu trong bài toán phân lớp dữ liệu mất cân bằng. Độ phức tạp tính toán của HBU tương đồng với HMU.

So với thuật toán HMU, HBU khắc phục nhược điểm của HMU trong các bài toán phân lớp khi các phần tử thuộc hai lớp đa số và thiểu số nằm gần nhau.

So sánh với thuật toán DEC-SVM thì thuật toán HBU cũng có các đặc điểm tương tự đã trình bày trong mục 2.2.3 ở trên.

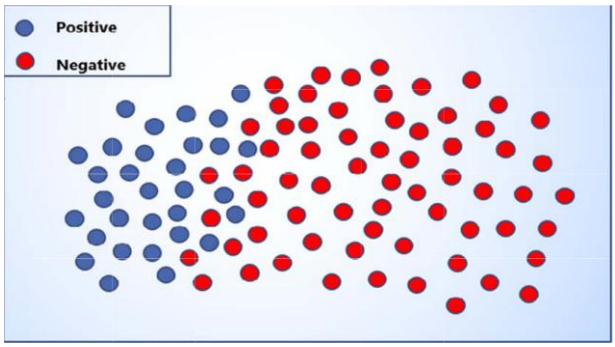
2.4. Thuật toán RBU

2.4.1. Giới thiệu thuật toán

Thuật toán RBU (Random Border Undersampling) cải tiến từ thuật toán Random undersampling có sẵn, sử dụng việc giảm ngẫu nhiên phần tử trên đường biên. Để xác định các phần tử trên đường biên, thuật toán xác định dựa vào số láng giềng là thuộc lớp thiểu số m trong tổng số k láng giềng gần nhất. Nếu $k/2 \leq m < k$ thì phần tử đó là phần tử biên.

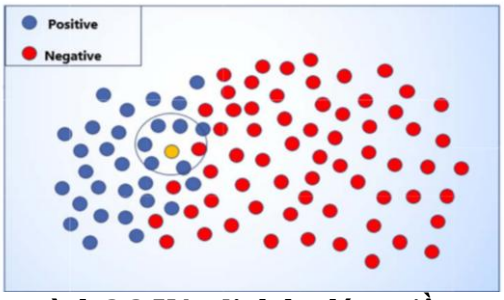
Dưới đây mô tả về bộ dữ liệu mất cân bằng và quá trình xác định phần tử lớp đa số thuộc đường biên để tiến hành giảm bớt các phần tử đó theo tỉ lệ phần trăm.

Hình 2.2 mô tả một vùng của bộ dữ liệu lớn. Có thể nhận thấy ngay sự chênh lệch về số phần tử giữa hai lớp đa số và lớp thiểu số.



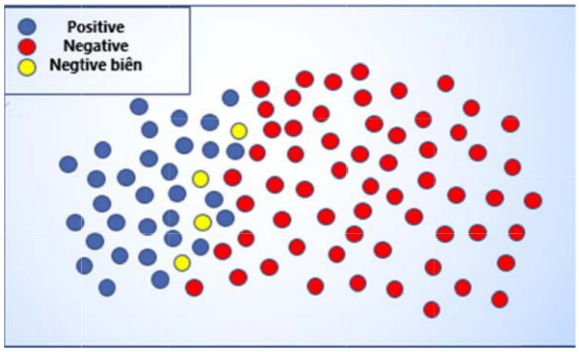
Hình 2.1 Phân bố dữ liệu

Hình 2.3 mô tả quá trình xác định k láng giềng cho từng phần tử lớp đa số. Giả sử trong k láng giềng gần nhất có m láng giềng là thuộc lớp thiểu số. Nếu m thỏa mãn **$k/2 \leq m < k$** thì phần tử thuộc lớp đa số đang xét là phần tử thuộc đường biên. Hình 2.3, phần tử đang xét là phần tử được đánh dấu màu vàng, trong 7 láng giềng gần nhất của nó có 6 phần tử thuộc lớp thiểu số, 1 thuộc lớp đa số. Vậy phần tử đó thuộc biên.

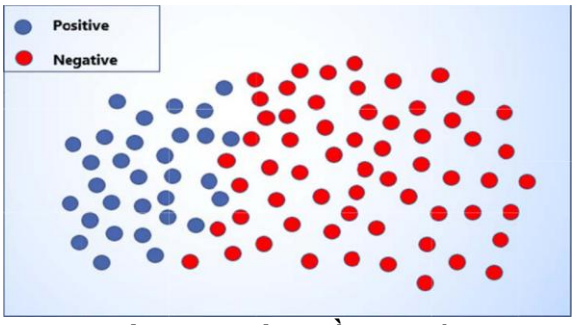


Hình 2.2 Xác định k - láng giềng

Tương tự, ta xác định được các phần tử lớp đa số thuộc đường biên là các phần tử màu vàng như trong hình 2.4.



Hình 2.3 Các phần tử biên



Hình 2.4 Xóa phần tử biên

Tiến hành xóa n% số phần tử biên thuộc lớp đa số đã xác định. Hình 2.5 là các phần tử thuộc biên đã bị xóa toàn bộ. Tuy nhiên, trong thuật toán mới, số phần tử biên sẽ bị xóa theo tỉ lệ phần trăm, phụ thuộc vào tham số n.

2.4.2. Khảo sát nội dung thuật toán

Thuật toán RBU có thể mô tả dưới dạng giả code như sau: ***RBU Algorthim***

Input: Bộ dữ liệu huấn luyện T gồm P positive lớp thiểu số, N negative lớp đa số

n: tỉ lệ phần trăm số phần tử trên biên bị giảm

k: số láng giềng gần nhất đối với một phần tử lớp đa số

m- số phần tử lớp đa số trên đườ ng biên

Output: Tập dữ liệu đã giảm các phần tử trên biên theo tỉ lệ phần trăm n

Các bước thực hiện của thuật toán:

Bước 1: Tìm các phần tử biên thuộc lớp đa số

Với mỗi phần tử trong tập lớp đa số N = {N_1, N_2, $N_{3...}$ N_i} tính k láng giềng gần nhất của nó trong toàn bộ tập dữ liệu huấn luyện T. Gọi số láng giềng thuộc lớp thiểu số trong tổng số k láng giềng gần nhất là m.

Bước 2: Xác định phần tử biên lớp đa số. Nếu **k/2 ≤ m < k** nghĩa là số láng giềng của N thuộc lớp thiểu số lớn hơn số láng giềng thuộc lớp đa số.

Bước 3: Đưa những phần tử thuộc N này vào một mảng border (mảng chứa các phần tử trên đường biên). Các phần tử trong mảng border là phần tử biên lớp đa số.

Bước 4: Giảm theo n phần trăm số phần tử trên đường biên để làm giảm tính mất cân bằng bộ dữ liệu.

Ghi chú:

- Trong thuật toán RBU, thông thường cho hai tham số đầu vào n chạy từ 1 đến 7 và k chạy từ 2 đến 10 để thuật toán khách quan và tổng quát hơn.

Tương tự các thuật toán HMU và HBU, trong pha tiền xử lý dữ liệu có thể sử dung RBU và sau đó sử dụng các thuật toán học máy truyền thống như SVM để xây dựng các bộ phân lớp dữ liệu đối với các dữ liệu mất cân bằng.

Có thể đề xuất thuật toán phân lớp dữ liệu RBU-SVM như sau:

```
Void RBU-SVM() {
        Thực hiện RBU tiền xử lý dữ liêu;
                SVM training;
                SVM prediction;
        }
```

2.4.3. Đánh giá thuật toán

Tương tự các thuật toán HMU và HBU, thuật toán RBU ứng dụng trong pha tiền xử lý dữ liệu trong bài toán phân lớp dữ liệu mất cân bằng. Các thuật toán HMU, HBU và RBU đều nhằm giảm số lượng phân tử của lớp đa số. Độ phức tạp tính toán của RBU tương đồng với HMU và HBU.

2.5. Kết luận chương 2

Trong chương 2, luận văn đã khảo sát một số thuật toán: DEC - SVM, HMU, HBU, RBU. Các thuật toán đều thực hiện các ý tưởng nhằm điều chỉnh dữ liệu của tập dữ liệu mất cân bằng trước khi tiến hành quá trình phân lớp dữ liệu. Điều đó sẽ giúp cho các mô hình phân lớp sử dụng các thuật toán học máy hiệu quả hơn.

Trong chương 3, luận văn sẽ ứng dụng các thuật toán cho bài toán phân lớp dữ liệu mất cân bằng đối với bộ dữ liệu thực tế.

CHƯƠNG 3. ỨNG DỤNG

3.1. Khảo sát và lựa chọn bộ dữ liệu để thử nghiệm

3.1.1. Giới thiệu

Bệnh tiểu đường trong thời đại hiện nay là một trong những căn bệnh phổ biến nhất. Ở một số nước, số người mắc căn bệnh này chiếm tỉ lệ tới 10% dân và số người mắc bệnh ngày một tăng cao. Phần lớn bệnh nhân mắc chứng tiểu đường type 2 và tỉ lệ người bệnh tăng cao liên quan trực tiếp với cách sống của cuộc sống hiện đại ngày nay.

Một thí dụ điển hình là số phận của những người Ấn Độ Pima. Các bác sỹ phát hiện ra rằng ở những người da đỏ này có gien tiềm ẩn bệnh tiểu đường type 2. Trong cơ thể những người da đỏ này có loại gien làm cho các tế bào kém nhậy cảm với insulin mà hậu quả của nó là các tế bào chuyển hóa lượng đường rất ít thành năng lượng.

Việc khám phá kiến thức từ cơ sở dữ liệu y tế là rất quan trọng để giúp chẩn đoán y tế hiệu quả. Trong luận văn này bộ dữ liệu được sử dụng là Bộ dữ liệu bệnh tiểu đường của người Ấn Độ Pima, thu thập thông tin của bệnh nhân mắc và không mắc bệnh tiểu đường.

3.1.2. Mô tả bộ dữ liệu Pima-indians-diabetes

Bộ dữ liệu bệnh tiểu đường của người Ấn Độ Pima, gồm 768 hồ sơ bệnh nhân nữ ít nhất 21 tuổi của người Ấn Độ Pima, một dân số sống gần Phoenix, Arizona, Hoa Kỳ.

Các thuộc tính của bộ dữ liệu Pima-indians-diabetes được mô tả dưới đây.

Bảng 3.1 Các thuộc tính của bộ dữ liệu Pima-indians-diabetes

TT	Tên thuộc tính	Mô tả	Tính chất
1	Pregnancies	Số lần mang thai	
2	Glucose	Nồng độ glucose huyết tương 2h trong xét nghiệm dung nạp glucose đường uống	
3	BloodPressure	Huyết áp tâm trương (mm Hg)	
4	Skinthickness	Độ dày nếp gấp da (mm)	
5	Insulin	Huyết thanh 2 giờ (mu U / ml)	
6	BMI	Chỉ số khối cơ thể	
7	Diabetespedigree	Chức năng phả hệ tiểu đường	
8	Age	Tuổi (năm).	
9	Outcome	Thuộc tính phân lớp (0,1)	

3.2. Xây dựng kịch bản và lựa chọn công cụ thử nghiệm

3.2.1. Xây dựng kịch bản thử nghiệm:

Dữ liệu đầu vào:

(1) Bộ dữ liệu ***pima-indians-diabetes***

(2) Các thuật toán thử nghiệm: DEC-SVM, HMU-SVM, HBU-SVM, RBU-SVM

Dữ liệu ra:

Các tiêu chí, kết quả đánh giá hiệu năng của các thuật toán nghiên cứu trong chương 2 áp dụng vớ bộ dữ liệu ***pima-indians-diabetes***

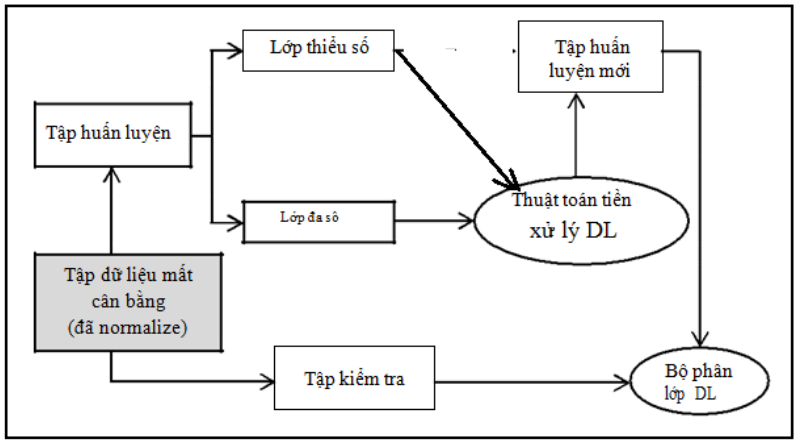
Luận văn sẽ tiến hành thử nghiệm theo hai kịch bản trình bày dưới đây.

Kịch bản thứ nhất: Kịch bản này, luận văn sử dụng thuật toán SVM để phân lớp dữ liệu với bộ dữ liệu đã chọn, không sử dụng các thuật toán tiền xử lý dữ liệu mất cân bằng.

Kịch bản thứ hai: Trong kịch bản thứ hai, luận văn sẽ thực hiện phân lớp dữ liệu sau khi xử lý dữ liệu mất cân bằng sử dụng thuật toán xử lý dữ liệu mất cân bằng.

3.2.2. Mô hình thử nghiệm

Mô hình tiến hành thử nghiệm được mô tả trong hình 3.1 dưới đây.



Hình 3.1 Mô hình thử nghiệm

Các thuật toán tiền xử lý dữ liệu lựa chọn lần lượt là DEC-SVM, HMU, HBU và RBU.

3.2.3. Lựa chọn công cụ thử nghiệm

Weka là một phần mềm miễn phí về học máy được viết bằng Java, luận văn lựa chọn công cụ thực nghiệm là phần mềm Weka version 3.7.12 [19].

Các tính năng chính của Weka:

- Weka bao gồm một tập các công cụ tiền xử lý dữ liệu, các thuật toán học máy để khai phá dữ liệu và các phương pháp thử nghiệm đánh giá.

- Weka có giao diện đồ họa (gồm cả tính năng hiển thị hóa dữ liệu)

- Weka bao gồm các môi trường cho phép so sánh các thuật toán học máy trên bộ dữ liệu do người dùng lựa chọn.

Các môi trường chính trong Weka:

(1) Simple CLI : giao diện đơn giản kiểu dòng lệnh (như MS-DOS).

(2) Explorer : môi trường cho phép sử dụng các khả năng của Weka để khám phá dữ liệu.

(3) Experimenter: môi trường cho phép tiến hành các thí nghiệm và thực hiện các kiểm tra thống kê (statistical tests) giữa các mô hình máy học. Môi trường này bao gồm:

(4) KnowledgerFlow: môi trường cho phép bạn tương tác đồ họa kiểu kéo/ thả để thiết kế các bước(các thành phần) của một thí nghiệm.

3.3. Thử nghiệm và đánh giá kết quả thử nghiệm

3.3.1. Mô tả thử nghiệm

Bộ dữ liệu thử nghiệm là *pima-indians-diabetes.csv* gồm 768 bản ghi, 9 thuộc tính.

Các thuật toán thử nghiêm: DEC-SVM, HBU, HMU, RBU

Các bước thực hiên như sau:

Bước 1: Chuẩn hóa dữ liệu bằng Filter standardize của Weka. Dữ liệu được xử lý để có kỳ vọng tại 0 và có độ lệch chuẩn bằng 1. Việc chuẩn hóa giúp các thuật toán không bị thiên lệch về một số đặc trưng và hơn nữa giúp quá trình học hội tụ nhanh hơn.

Đối với kịch bản 2: thực hiện bước 2

Bước 2: Cân bằng dữ liệu bằng một trong các thuật toán đề xuất (RBU, HBU, HMU, DEC-SVM)

Bước 3: Với bộ dữ liệu thu được, phân lớp bằng thuật toán SVM trong Weka

3.3.2. Kết quả thử nghiệm

(1) Kết quả phân lớp trước khi xử lý dữ liệu mất cân bằng theo kịch bản 1

Bảng 3.2 Kết quả phân lớp trước khi xử lý dữ liệu mất cân bằng sử dụng thuật toán SVM

```
=== Detailed Accuracy By Class ===
TP Rate   FP Rate   Precision   Recall  F-Measure  ROC Area  Class
0.866     0.448      0.783     0.866     0.822      0.709    0
0.552     0.134      0.688     0.552     0.613      0.709    1
0.757     0.338      0.75      0.757     0.749      0.709    Avg.
=== Confusion Matrix ===
   a   b   <-- classified as
 433  67 |   a = 0
 120 148 |   b = 1
```

(2) Kết quả phân lớp sau khi xử lý dữ liệu mất cân bằng theo kịch bản 2

Bảng 3.3 Kết quả phân lớp sau khi xử lý dữ liệu mất cân bằng với thuật toán DEC-SVM

```
=== Detailed Accuracy By Class ===
TP Rate   FP Rate   Precision   Recall  F-Measure  ROC Area  Class
0.757     0.213      0.775     0.757     0.766      0.772    0
0.787     0.243      0.77      0.787     0.778      0.772    1
0.772     0.228      0.772     0.772     0.772      0.772    Avg.
=== Confusion Matrix ===
   a   b   <-- classified as
 355 114 |   a = 0
 103 381 |   b = 1
```

Bảng 3.4 Kết quả phân lớp sau khi xử lý dữ liệu mất cân bằng với thuật toán HMU

```
=== Detailed Accuracy By Class ===
TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area Class
0.741     0.187      0.8       0.741     0.769      0.777    0
0.813     0.259      0.757     0.813     0.784      0.777    1
0.777     0.223      0.779     0.777     0.777      0.777    Avg.
=== Confusion Matrix ===
   a   b   <-- classified as
 200  70 |   a = 0
  50 218 |   b = 1
```

Bảng 3.5 Kết quả phân lớp sau khi xử lý dữ liệu mất cân bằng với thuật toán HBU

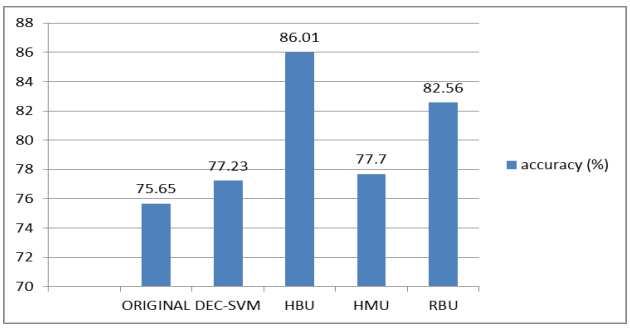
```
=== Detailed Accuracy By Class ===
TP Rate   FP Rate   Precision  Recall  F-Measure   ROC Area  Class
0.866     0.146      0.856     0.866     0.861      0.86     0
0.854     0.134      0.864     0.854     0.859      0.86     1
0.86      0.14       0.86      0.86      0.86       0.86     Avg.
=== Confusion Matrix ===
   a   b   <-- classified as
 232  36 |   a = 0
  39 229 |   b = 1
```

Bảng 3.6 Kết quả phân lớp sau khi xử lý dữ liệu mất cân bằng với thuật toán RBU

```
=== Detailed Accuracy By Class ===
TP Rate   FP Rate   Precision  Recall  F-Measure  ROC Area  Class
0.906     0.302      0.827     0.906     0.865      0.802    0
0.698     0.094      0.824     0.698     0.756      0.802    1
0.826     0.222      0.825     0.826     0.822      0.802    Avg.
=== Confusion Matrix ===
   a   b   <-- classified as
 386  40 |   a = 0
  81 187 |   b = 1
```

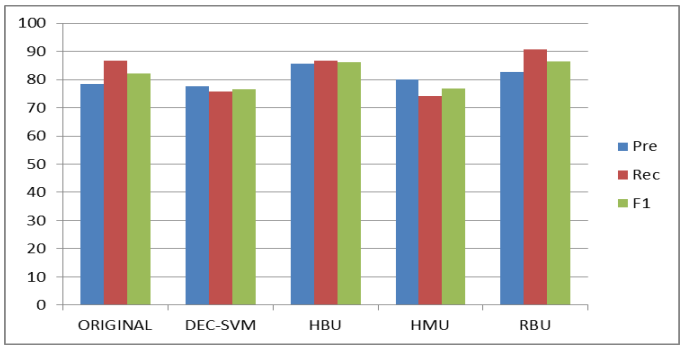
3.3.3. Đánh giá kết quả thử nghiệm

Dựa vào kết quả thử nghiệm đã trình bày ở mục trên, mục này luận văn sẽ thực hiện phân tích và đánh giá kết quả. Kết quả độ chính xác của các thuật toán thử nghiệm theo hai kịch bản được biểu diễn dưới dạng biểu đồ như trong hình 3.3.

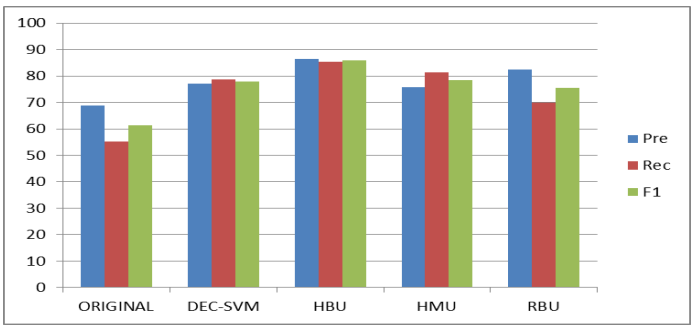


Hình 3.2 Biểu đồ so sánh độ chính xác phân lớp trên dữ liệu trước và sau khi xử lý dữ liệu.

Quan sát biểu đồ nhận thấy rằng, các thuật toán thử nghiệm đều cho kết quả có tỉ lệ phân loại chính xác cao hơn so với bộ dữ liệu ban đầu khi chưa áp dụng thuật toán.



Hình 3.3 Biểu đồ kết quả phân lớp lớp Negative trước và sau khi xử lý dữ liệu mất cân bằng



Hình 3.4 Biểu đồ kết quả phân lớp lớp Positive trước và sau khi xử lý dữ liệu mất cân bằng

Từ các kết quả ở trên ta thấy sau khi điều chỉnh bộ dữ liệu bởi các thuật toán tiền xử lý dữ liệu mất cân bằng DEC-SVM, HBU, HMU, RBU thì hiệu quả phân lớp các bộ dữ liệu cao hơn hẳn so với việc phân lớp của bộ dữ liệu ban đầu.

3.4. Kết luận chương 3

Trong chương 3 luận văn đã tiến hành thử nghiệm các thuật toán DEC-SVM, HMU, HBU và RBU cho bài toán phân lớp dữ liệu trên dữ liệu mất cân bằng cho bộ dữ liệu về chứng tiểu đường của người Indian Pima.

Kết quả thử nghiệm bước đầu cho thấy các thuật toán phân lớp trên có thể triển khai trong thực tế và phù hợp các yêu cầu đề ra cho bài toán phân lớp dữ liệu trên dữ liệu mất cân bằng

KẾT LUẬN

Kết quả đạt được của luận văn

Với mục tiêu nghiên cứu một số kỹ thuật để nâng cao hiệu năng phân lớp dữ liệu trên tập dữ liệu mất cân bằng và ứng dụng, luận văn đã đạt được một số kết quả như sau:

- Nghiên cứu tổng quan về bài toán phân lớp dữ liệu và các vấn đề liên quan.
- Khảo sát tổng quan về dữ liệu mất cân bằng.
- Khảo sát hướng tiếp cận về dữ liệu và hướng tiếp cận về thuật toán để nâng cao hiệu năng phân lớp dữ liệu trên dữ liệu mất cân bằng.
- Khảo sát chi tiết các thuật toán: DEC-SVM, HMU, HBU và RBU.
- Khảo sát bộ dữ liệu về bệnh tiểu đường pima-indians-diabetes.
- Thực hiện thử nghiệm phân lớp dữ liệu với DEC-SVM, HMU, HBU và RBU trên bộ dữ liệu pima-indians-diabetes. Kết quả thử nghiệm cho thấy hiệu quả phân lớp dữ liệu sau khi sử dụng thuật toán đã được khảo sát.

Tuy nhiên, do hạn chế về mặt thời gian, luận văn chưa tiến hành thử nghiệm với các bộ dữ liệu lớn, Do đó, hiệu quả thử nghiệm chưa cao.

Hướng phát triển tiếp theo

Trên cơ sở nghiên cứu và các kết quả đạt được, đề tài luận văn có thể phát triển tiếp theo như sau:

- Tiếp tục hoàn thiện các kết quả đã có để có thể xây dựng các mô hình phân lớp trên dữ liệu mất cân bằng với các bộ dữ liệu trong thực tế thường có kích thước lớn, các thuộc tính của các phần tử dữ liệu thường bao gồm cả dạng số và dạng phi số.
- Nghiên cứu thêm về các kỹ thuật trích chọn đặc trưng cho các bộ dữ liệu mất cân bằng nhằm nâng cao hiệu quả cho các mô hình phân lớp.