ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

PHẠM VĂN HIẾU

DỰ ĐOÁN TƯƠNG TÁC PROTEIN - PROTEIN SỬ DỤNG KỸ THUẬT KHAI PHÁ DỮ LIỆU

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

Hà Nội – 2017

ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

PHẠM VĂN HIẾU

DỰ ĐOÁN TƯƠNG TÁC PROTEIN – PROTEIN SỬ DỤNG KỸ THUẬT KHAI PHÁ DỮ LIỆU

NGÀNH: CÔNG NGHỆ THÔNG TIN

CHUYÊN NGÀNH: HỆ THỐNG THÔNG TIN

MÃ SỐ: 60480104

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. ĐẶNG THANH HẢI

Hà Nội – 2017

LỜI CAM ĐOAN

Tôi xin cam đoan nội dung của luận văn "***Dự đoán tương tác protein – protein sử dụng kỹ thuật khai phá dữ liệu***" là sản phẩm do tôi thực hiện dưới sự hướng dẫn của **TS. Đặng Thanh Hải**. Trong toàn bộ nội dung của luận văn, những điều được trình bày là do tôi nghiên cứu được từ các tài liệu tham khảo. Tất cả các tài liệu tham khảo đều có xuất xứ rõ ràng và được trích dẫn hợp pháp.

Tôi xin chịu trách nhiệm cho lời cam đoan của mình.

Hà Nội, ngày 10 tháng 10 năm 2017

Người cam đoan

Phạm Văn Hiếu

LỜI CẢM ƠN

Tôi xin bày tỏ lòng biết ơn sâu sắc đến thầy hướng dẫn của tôi, TS. Đặng Thanh Hải. Thầy đã giúp tôi có những cơ hội để có thể theo đuổi nghiên cứu lĩnh vực mình yêu thích. Trong suốt quá trình thực hiện luận văn, thầy đã tận tình hướng dẫn cho tôi, góp ý cho tôi về đường lối, đồng thời đưa ra những lời khuyên bổ ích để tôi có thể hoàn thành luận văn của mình.

Tiếp đến, tôi xin chân thành cảm ơn các thầy cô giáo trong Khoa Công nghệ Thông tin, Đại học Công nghệ - Đại học Quốc gia Hà Nội đã truyền đạt cho tôi những kiến thức và kinh nghiệm vô cùng quí báu trong quá trình học tập và nghiên cứu.

Tôi cũng muốn cảm ơn các bạn cùng lớp và các đồng nghiệp đã cho tôi những lời động viên, những hỗ trợ và góp ý về mặt chuyên môn.

Cuối cùng, tôi xin cảm ơn gia đình, bạn bè, những người đã luôn bên cạnh ủng hộ và động viên tôi.

Hà Nội, tháng 10 năm 2017

Phạm Văn Hiếu

MỤC LỤC

DANH MỤC HÌNH VẼ VÀ ĐỒ THỊ

DANH MỤC BẢNG BIỂU

CHƯƠNG 1 : MỞ ĐẦU

1.1 LÝ DO CHỌN ĐỀ TÀI

Protein là thành phần quan trọng trong tế bào nói riêng và cơ thể sống nói chung, và tương tác protein – protein là một cách để các protein thể hiện được các chức năng sinh học của mình. Vì vậy hiểu biết về các tương tác protein – protein sẽ giúp chúng ta hiểu sâu hơn về các chức năng protein, và tìm ra được vai trò của các protein mới.

Vào thời điểm bắt đầu nghiên cứu về tương tác protein – protein, các nhà khoa học thường sử dụng phương pháp hóa sinh để phân tích và dự đoán. Tuy nhiên các phương pháp thực nghiệm này đắt tiền, tốn nhiều thời gian, công sức, và nhiều khi rất khó để thực hiện. Vì vậy nên yêu cầu cấp thiết được đặt ra là dự đoán bằng cách áp dụng khai phá dữ liệu và phát triển các mô hình tính toán tự động để đạt hiệu quả cao, nhanh hơn như là sự bổ sung cho các phương pháp thực nghiệm.

Theo thời gian, số lượng ngày càng tăng của tập các cặp protein – protein tương tác với nhau (và tập không tương tác) đã được thực nghiệm xác định. Sự tích lũy dữ liệu về tương tác protein – protein bằng thực nghiệm đem lại lợi thế về mặt đầy đủ thông tin để có thể tính toán dự đoán được thêm các tương tác protein – protein mới. Và đó cũng là lý do tôi quyết định chọn đề tài **"Dự đoán tương tác protein – protein sử dụng kỹ thuật khai phá dữ liệu"**.

1.2 MỤC TIÊU ĐỀ TÀI

Trong khuôn khổ luận văn này, tôi trình bày một phương pháp tính toán cho dự đoán tương tác protein – protein khác với các phương pháp phân lớp truyền thống, đó là xây dựng mô hình phân lớp theo hướng áp dụng thuật toán phân lớp tổng hợp, hay là sự kết hợp mô hình các bộ phân lớp đơn lẻ yếu hơn thành một mô hình mạnh, nhằm đạt được hiệu quả phân lớp tối ưu.

Với bài toán như trên, đặt ra mục tiêu cho đề tài là tìm hiểu và xây dựng thành công một mô hình dự đoán tương tác protein-protein dựa trên thuật toán phân lớp tổng hợp, là phương pháp đã được chứng minh là tốt hơn thuật toán phân lớp đơn lẻ truyền thống, từ đó làm tiền đề áp dụng vào thực tế triển khai nghiên cứu dự đoán tương tác protein – protein một cách hiệu quả nhất. Để đạt được mục tiêu đó, các công việc tôi đã thực hiện trong luận văn này là: Nghiên cứu cơ sở lý thuyết các khái niệm về protein, cấu trúc protein trong sinh học, nhằm phục vụ cho việc khai thác các thuộc tính của chúng sử dụng trong tính toán; Nghiên cứu cơ sở lý thuyết về các kỹ thuật khai phá dữ liệu (nói

chung) và kỹ thuật phân lớp dữ liệu (nói riêng), làm cơ sở cho xây dựng chương trình thực nghiệm và chứng minh tính đúng đắn của kết quả thực nghiệm.

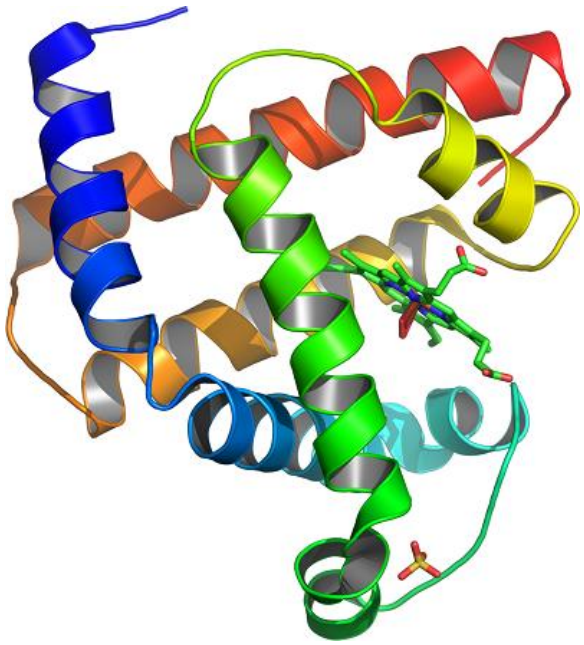
Với chương trình thực nghiệm, bước đầu tôi đã đạt được mục tiêu của đề tài là chứng minh được tính hiệu quả khi áp dụng giải thuật phân lớp tổng hợp vào bài toán dự đoán tương tác protein – protein so với các giải thuật khác. Qua đó có thể đạt được những mục tiêu xa hơn trong tương lai, ví dụ như từ giải thuật trong đề tài này có thể làm nền móng cho các giải thuật khác triển khai hiệu quả hơn, giúp tăng hiệu năng cũng như độ chính xác của bài toán "Dự đoán tương tác protein – protein sử dụng kỹ thuật khai phá dữ liệu".

CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT

Chương 2 trình bày cơ sở lý thuyết, bao gồm các thông tin giới thiệu về các khái niệm trong sinh học liên quan đến protein, cấu trúc protein; Các khái niệm khai phá dữ liệu nền tảng liên quan đến kỹ thuật phân lớp dữ liệu, nhằm củng cố kiến thức và tạo tiền đề áp dụng giải quyết bài toán "Dự đoán tương tác protein – protein sử dụng kỹ thuật khai phá dữ liệu".

2.1 CÁC KHÁI NIỆM LIÊN QUAN ĐẾN PROTEIN

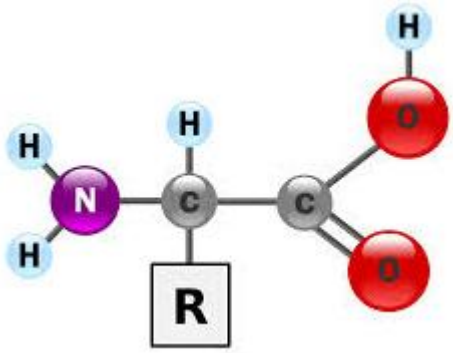
Protein là đại phân tử, phức tạp và có vai trò quan trọng trong tế bào (nói riêng) và cơ thể sống (nói chung). Chúng được tạo thành từ hàng trăm hoặc hàng ngàn các đơn vị nhỏ hơn được gọi là các amino acid. Protein được tạo ra bởi sự liên kết của hai hoặc nhiều polypeptide, là chuỗi được ghép từ các amino acid liên kết với nhau, được xếp thành một cấu trúc đặc biệt cho mỗi một protein cụ thể [1].



Hình 2-1: Minh họa cấu trúc 3D một protein *[2]*

2.1.1 Cấu trúc Protein

Protein được hình thành do các amino acid liên kết lại với nhau bởi các liên kết peptide tạo ra chuỗi polypeptide. Amino acid được cấu tạo bởi 3 thành phần : nhóm amin ($-NH_2$), nhóm caboxyl ($-COOH$) và cuối cùng là nguyên tử cacbon trung tâm đính với 1 nguyên tử hydro và nhóm biển đổi R quyết định tính chất của amino acid.



Hình 2-2: Cấu tạo của một amino acid

Có tất cả 20 loại amio acid trong thành phần của tất cả các loại protein khác nhau. Nhưng dựa vào cấu tạo gốc R chúng ta có thể phân lớp tổng quan thành 5 nhóm có các tính chất hóa lý đặc trưng riêng, cụ thể:

- Các amio acid có gốc R không phân cực, kị nước (Glycine, Alanine, Valine, Leucine, Isoleucine, Proline).
- Các amio acid có gốc R là nhân thơm (Phenylalanine, Tyrosine, Tryptophan).
- Các amio acid có gốc R bazơ, tích điện dương (Lysine, Arginine, Histidine).
- Các amio acid có gốc R phân cực, không tích điện (Serine, Threonine, Cysteine, Methionine, Asparagine, Glutamine).
- Các amio acid có gốc R acid, tích điện âm (Aspartate, Glutamate).

Phân tử protein thường được chia làm hai dạng: Protein hình cầu và protein dạng sợi. Các protein hình cầu có đặc điểm chung là nhỏ gọn, dễ hòa tan và dạng hình cầu. Protein dạng sợi thường kéo dài và không hòa tan. Các đặc tính này phụ thuộc vào cấu trúc mà protein đó quy định. Các loại cấu trúc này gồm có: Cấu trúc sơ cấp, cấu trúc bậc hai, cấu trúc bậc ba, cấu trúc bậc bốn [3]. Cụ thể:

- Cấu trúc sơ cấp: Là cấu trúc mô tả thứ tự mà trong đó các amino acid được liên kết với nhau để tạo thành một protein. Thứ tự của các amino acid trong một chuỗi polypeptide là duy nhất và riêng biệt cho mỗi protein riêng biệt. Thay đổi một acid amin đơn lẻ có thể gây ra đột biến gene, thường dẫn đến một protein không thực hiện được chức năng vốn có.
- Cấu trúc bậc hai: Là cấu trúc đề cập đến việc xoắn hoặc gấp một chuỗi polypeptide cho protein hình dạng 3D của nó. Có hai loại cấu trúc bậc 2 quan sát được trong các protein. Một loại là cấu trúc xoắn alpha (α), cấu trúc này giống như một lò xo xoắn và được bảo vệ bởi liên kết hydro trong chuỗi polypeptide.

Loại thứ hai là cấu trúc nếp gấp Beta (β), cấu trúc này trông như các nếp gấp lại và được giữ bởi các liên kết hydro giữa các đơn vị polypeptide của chuỗi gấp xếp liền kề nhau.

- Cấu trúc bậc ba : Là cấu trúc đề cập đến cấu trúc 3-D toàn diện của chuỗi polypeptide của một protein. Có một số loại liên kết và lực giữ một protein trong cấu trúc bậc ba của nó. Những tương tác liên quan đến các lực hấp dẫn xảy ra giữa các phân tử bị phân cực. Những lực này đóng góp vào sự liên kết xảy ra giữa các phân tử.
- Cấu trúc bậc bốn : Đề cập đến cấu trúc của một phân tử protein được hình thành bởi các tương tác giữa nhiều chuỗi polypeptide. Mỗi chuỗi polypeptide được coi như một đơn vị con. Protein có cấu trúc bậc bốn có thể bao gồm nhiều hơn một loại đơn vị con protein giống nhau. Ví dụ như hemoglobin được tìm thấy trong máu, bao gồm bốn tiểu đơn vị: hai tiểu đơn vị alpha (α) và hai tiểu đơn vị Beta (β).

2.1.2 Chức năng của Protein

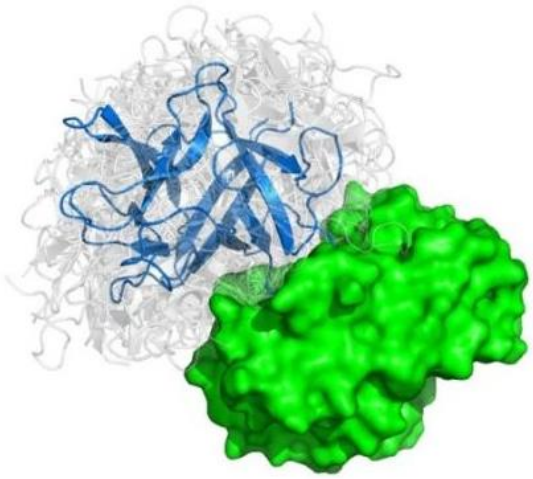
Protein đảm nhiệm các chức năng liên quan đến toàn bộ hoạt động sống của tế bào, quy định các tính trạng và các tính chất của cơ thể sống. Cụ thể:

Bảng 2-1: Bảng chức năng các loại protein cơ bản *[4]*

Loại protein	Chức năng
Protein vận động	Chịu trách nhiệm cho sự co cơ và chuyển động.
Protein cấu trúc	Có tính chất xơ và bền nên có ý nghĩa cung cấp sự hỗ trợ cho các bộ phận khác nhau của cơ thể
Protein Enzyme	Giúp tạo ra các phản ứng sinh hóa. Thường được gọi là chất xúc tác vì chúng đẩy nhanh các phản ứng hóa học.
Protein Hormone	Giúp điều hòa các hoạt động sinh lý trong cơ thể.
Protein vận chuyển	Chịu trách nhiệm vận chuyển các chất từ nơi này đến nơi khác trong cơ thể.
Protein kháng thể	Có vai trò bảo vệ cơ thể khỏi các kháng nguyên xâm nhập.
Protein dự trữ	Có vai trò dự trữ chất dinh dưỡng cho cơ thể

2.1.3 Định nghĩa quan hệ tương tác protein – protein (PPI)

Tương tác protein – protein là quá trình tác động qua lại giữa các protein với nhau trong tế bào ảnh hưởng đến các hoạt động sống của tế bào và ảnh hưởng đến quá trình sống của động vật. Về mặt vật lý, tương tác protein – protein là hiện tượng hai hay nhiều protein bám vào nhau trong một điều kiện sinh hóa cụ thể dưới tác động của lực hút tĩnh điện và ảnh hưởng của tính kỵ nước của protein để tạo thành phức hợp cùng tham gia vào một quá trình sinh học nào đó.



Hình 2-3: Minh họa tương tác protein – protein *[5]*

Các loại tương tác protein – protein bao gồm :

- Tương tác ổn định
- Tương tác tạm thời
- Tương tác mạnh
- Tương tác yếu

2.1.4 Tầm quan trọng của tương tác protein – protein

Sự tương tác của protein – protein là nền tảng cơ bản của các chức năng của tế bào và khi quá trình tương tác này bị tổn hại sẽ gây ảnh hưởng trực tiếp đến cơ thể sống [6]. Các ảnh hưởng sinh học của quá trình tương tác protein – protein tác động tới cơ thể sống là:

- Thay đổi các tính chất động học của enzyme : có thể trong liên kết cấu trúc hoặc các ảnh hưởng allosteric.
- Tạo các điểm liên kết mới.

- Bất hoạt hoặc phá hủy một protein.
- Thay đặc tính của một protein.
- Điều tiết các quá trình.
- Tạo các kênh cơ chất bằng việc di chuyển cơ chất giữa các vùng hoặc các tiểu đơn vị.

2.2 KHÁI NIỆM CƠ BẢN VỀ KHAI PHÁ DỮ LIỆU

2.2.1 Định nghĩa về khai phá dữ liệu

Khai phá dữ liệu là một lĩnh vực đa ngành. Nó dựa trên kết quả từ trí thông minh nhân tạo, xác suất và thống kê, lý thuyết tính toán phức tạp, lý thuyết kiểm soát, lý thuyết thông tin, triết học, tâm lý, thần kinh học và các lĩnh vực khác. Nó cho phép chương trình "học tập" và tự động cải thiện năng lực từ kinh nghiệm tích lũy [7]. Ví dụ như trong đề tài này, chương trình có thể "học" cách phân lớp một mối quan hệ protein – protein có phải là mối quan hệ tương tác hay không và tự động xếp chúng vào nhóm protein - protein tương tác (PPIs) hoặc nhóm protein – protein không tương tác (PPNIs).

Các thuật toán khai phá dữ liệu thường được chia thành hai loại tùy theo cách sử dụng chúng : Thuật toán học máy – có giám sát (phân lớp), và thuật toán học máy – không giám sát (phân cụm).

2.2.2 Định nghĩa về học có giám sát

Học có giám sát có mục đích là xây dựng một mô hình dự đoán dựa trên bằng chứng trong một trường hợp không chắc chắn. Thuật toán học có giám sát lấy một tập dữ liệu đầu vào đã biết kết quả đầu ra, và xây dựng một mô hình để tạo ra các dự đoán hợp lý cho kết quả của một dữ liệu mới. Học có giám sát sử dụng sử dụng các kỹ thuật phân lớp và hồi quy để phát triển các mô hình dự đoán.

Biểu diễn theo toán học, giả sử chúng ta có một tập hợp dữ liệu đầu vào $X = \{x_1, x_2, \dots, x_n\}$ đã biết kết quả phân lớp là $Y = \{y_1, y_2, \dots, y_n\}$. Học có giám sát là từ tập dữ liệu đầu vào dùng training tạo ra một hàm ánh xạ mỗi phần tử từ tập X sang phần tử tương ứng của tập Y:

$$y_i \approx f(x_i), \forall i = 1, 2, \dots n \quad (2.1)$$

Hàm ánh xạ này đóng vai trò là một mô hình, dùng trong trường hợp có dữ liệu đầu vào mới qua mô hình sẽ tính được kết quả phân lớp tương ứng với dữ liệu đầu vào. Ví dụ trong đề tài này ta có tập dữ liệu đầu vào là các cặp protein – protein đã gán nhãn kết

quả đầu ra là tương tác hoặc không tương tác. Sau khi thuật toán tạo ra một mô hình, tức là một hàm số mà đầu vào là một dữ liệu quan hệ protein – protein và đầu ra là một nhãn tương tác, hoặc không tương tác, khi nhận được một quan hệ protein – protein mới mà mô hình chưa nhìn thấy bao giờ, nó sẽ dự đoán được quan hệ đó là tương tác hay không tương tác.

Bảng 2-2: Bộ dữ liệu huấn luyện dự đoán tương tác PPI

PPI	Fea_1	Fea_2	Fea_3	Fea_4	…	Fea_m	Label
M_1	1.12E-4	2.64E-4	3.01E-4	1.13E-4	…	6.18E-4	1
M_2	1.11E-4	1.58E-4	2.57E-4	9.6E-5	…	4.77E-4	1
M_3	1.03E-4	2.46E-4	8.35E-4	0.0	…	6.39E-4	0
M_4	1.68E-4	2.01E-4	2.55E-4	2.55E-4	…	2.19E-4	1
M_5	9.3E-5	1.11E-4	3.35E-4	1.67E-4	…	2.16E-4	0
…	…	…	…	…	…	…	…
M_{n-5}	1.05E-4	6.2E-5	1.86E-4	6.2E-5	…	3.09E-4	0
M_{n-4}	1.01E-4	0.0	1.93E-4	0.0	…	2.71E-4	0
M_{n-3}	1.24E-4	7.8E-5	6.47E-4	4.13E-4	…	4.57E-4	1
M_{n-2}	1.43E-4	2.29E-4	6.71E-4	4.03E-4	…	1.62E-4	0
M_{n-1}	8.9E-5	1.71E-4	7.4E-5	7.4E-5	…	2.46E-4	1
M_n	1.58E-4	2.07E-4	3.8E-5	3.3E-4	…	3.59E-4	1
T_k	1.06E-4	1.67E-4	2.89E-4	1.45E-4	…	5.78E-4	?

2.2.3 Khái niệm về thuật toán phân lớp trong học có giám sát

Phân lớp là cách thức xử lý nhằm xếp các mẫu dữ liệu chưa biết vào một trong các lớp đã được định nghĩa trước. Các mẫu dữ liệu chưa biết này được xếp lớp dựa trên giá trị các thuộc tính của mẫu dữ liệu đó. Hay đặc trưng của mỗi lớp là tập các thuộc tính các mẫu dữ liệu được xếp trong lớp đó.

Các thuật toán phân lớp tiêu biểu gồm có: Cây quyết định, mạng Bayes, SVM, … Các thuật toán này xây dựng những mô hình có khả năng phân lớp cho một mẫu dữ liệu mới chưa biết dựa vào những mẫu tương tự đã học trước đó.

2.2.4 Bài toán phân lớp

Một bài toán phân lớp bao gồm 3 bước sau:

- Chuẩn bị dữ liệu
- Xây dựng mô hình từ tập dữ liệu huấn luyện
- Kiểm tra và đánh giá kết quả

Chuẩn bị dữ liệu: Bước này chúng ta chuẩn hóa dữ liệu về dạng cấu trúc mà bài toán phân lớp xử lý được, là dữ liệu dưới dạng bảng gồm 2 cột đối tượng và thuộc tính của đối tượng. Ở bước này chúng ta cũng thực hiện trích xuất các thuộc tính đặc trưng nhất trong tập các thuộc tính của bộ dữ liệu.

Xây dựng mô hình từ tập dữ liệu huấn luyện: Nhằm xây dựng một mô hình xác định một tập các lớp dữ liệu. Mô hình này được xây dựng bằng cách phân tích một tập dữ liệu huấn luyện (training dataset) có nhiều mẫu, trong đó mỗi mẫu dữ liệu được xác định bởi giá trị của các thuộc tính và đã thuộc về một trong các lớp đã đươc định nghĩa trước, biểu diễn bằng thuộc tính phân lớp. Để đảm bảo tính khách quan, chúng ta có thể tạo ra nhiều bộ dữ liệu huấn luyện, và mỗi bộ dữ liệu sẽ chọn ngẫu nhiên các mẫu dữ liệu huấn luyện từ một kho các mẫu.

Kiểm tra và đánh giá kết quả: Cần chuẩn bị một tập dữ liệu kiểm định có các phần tử không thuộc tập dữ liệu huấn luyện, đảm bảo cho kết quả đánh giá khách quan. Đưa các mẫu thuộc tập dữ liệu kiểm định qua mô hình phân lớp đã được xây dựng ở bước 2 để thu được kết quả dự đoán. So sánh kết quả dự đoán với kết quả phân lớp đúng của các mẫu dữ liệu kiểm định. Kết quả ta có độ chính xác của một mô hình phân lớp dựa trên tập dữ liệu kiểm định là tỷ lệ những mẫu dữ liệu kiểm định được phân lớp đúng bởi mô hình phân lớp đó.

2.2.5 Tổng quan về một số thuật toán phân lớp cơ bản

a, Naïve Bayes

Naïve Bayes là phương pháp phân lớp dựa vào thống kê theo định lý của Bayes, với giả thiết đặt ra rằng giá trị giữa các thuộc tính là độc lập với nhau. Naïve Bayes được nghiên cứu rộng rãi từ những năm 1950 và trong thực tế, nó đã chứng tỏ được hiệu quả trong nhiều ứng dụng liên quan, bao gồm phân lớp văn bản, chẩn đoán y tế và quản lý hiệu năng hệ thống [8].

Các bước thực hiện thuật toán Bayes:

- Bước 1: Huấn luyện Naïve Bayes (dựa vào tập dữ liệu)
  - Tính xác suất $P(C_i)$

- Tính xác suất $P(\frac{x_k}{C_i})$

- Bước 2: Mẫu dữ liệu mới được gán vào lớp có giá trị lớn nhất theo công thức:

$$\max(P(C_i)\prod_{k-1}^{n} P(\frac{x_k}{C_i})) \quad (2.2)$$

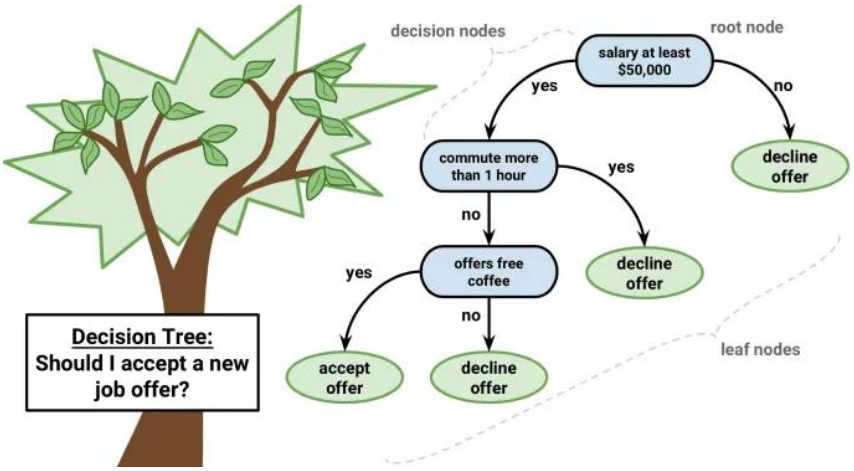
b, Cây quyết định

Cây quyết định (Decision Tree) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng chưa biết dựa trên các thuộc tính của đối tượng đó theo dãy các luật sinh ra từ một tập dữ liệu huấn luyện đã phân lớp. Hay các quy tắc xây dựng từ các thuộc tính của bộ dữ liệu huấn luyện được sử dụng để thực hiện dự đoán trên tập dữ liệu cần kiểm tra.

Hình dạng của một cây quyết định là một cấu trúc có thành phần: có node trên cùng được gọi là gốc, đó là thuộc tính có giá trị là điểm chia phân lớp tốt nhất trong tất cả các thuộc tính, các node ngoài cùng là các lá của cây quyết định, biểu thị cho các lớp đích biết trước mà đối tượng sẽ xếp vào. Giữa các node là các nhánh cây, đóng vai trò là các biểu thức so sánh để phân chia lớp của thuộc tính. Đường đi từ gốc đến lá cây là một chuỗi các quy tắc phân chia của giá trị thuộc tính, nếu thuộc tính của đối tượng chưa biết tuân theo các quy tắc này, sẽ quyết định đối tượng đó được xếp vào lớp có vị trí là node lá tận cùng của đường đi.

Cơ sở toán học của cây quyết định là thuật toán tham lam, trong đó các thuật toán xây dựng cây quyết định tiêu biểu là ID3, C4.5 và CART.

Cây quyết định là một phương pháp phân lớp hiệu quả và dễ hiểu, và được ứng dụng trong nhiều lĩnh vực như tài chính, tiếp thị, kỹ thuật và y học [9].

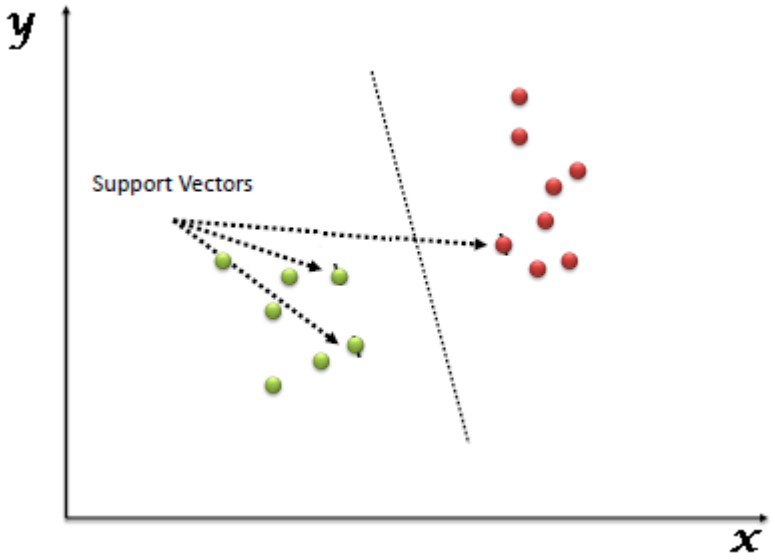


Hình 2-4: Minh họa Decision Tree

c, Support Vector Machine (SVM)

SVM là một thuật toán phân lớp nhị phân, SVM nhận dữ liệu vào và phân lớp chúng vào hai lớp khác nhau. Với một bộ các mẫu huấn luyện thuộc hai lớp cho trước, thuật toán SVM xây dựng một mô hình SVM để phân lớp các mẫu dữ liệu chưa biết vào hai lớp đó.

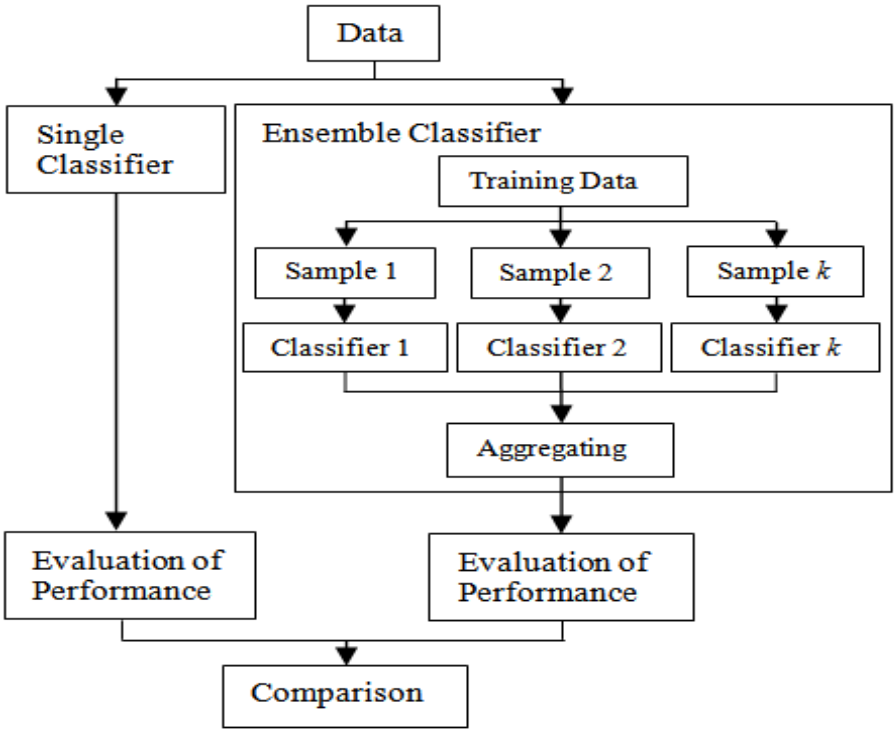
SVM thường cho độ chính xác cao đối với tập dữ liệu có kiểu dữ liệu liên tục.



Hình 2-5: Minh họa thuật toán SVM

2.2.6 Kết hợp các bộ phân lớp

Phương pháp phân lớp tổng hợp (ensemble) là mô hình có kết quả được tổng hợp từ nhiều mô hình con yếu (weaker model) được huấn luyện độc lập. Kết quả dự đoán cuối cùng dựa trên việc "bỏ phiếu" theo các kết quả của từng mô hình con đó để cho kết quả đầu ra. Các phân lớp con trong bộ phân lớp tổng hợp có thể là một bộ phân lớp truyền thống như: cây quyết định, mạng Bayes, ... Phương pháp phân lớp tổng hợp thường tạo ra các dự đoán chính xác hơn so với các phương pháp phân lớp đơn lẻ, do giảm ảnh hưởng từ quyết định mang tính tiên đoán khi chỉ có duy nhất một mô hình, từ đó giúp tạo ra các kết quả có độ chính xác được cải thiện.



Hình 2-6: So sánh bộ phân lớp đơn lẻ và bộ phân lớp tổng hợp

Có 2 phương pháp xây dựng một bộ phân lớp tổng hợp:

- Xây dựng mỗi bộ phân lớp cơ bản bên trong một cách độc lập, bằng cách thay đổi tập dữ liệu huấn luyện đầu vào, hoặc thay đổi các thuộc tính đặc trưng trong tập huấn luyện, sau đó sử dụng phương pháp biểu quyết để chọn ra kết quả cuối cùng của bộ phân lớp.
- Xây dựng các bộ phân lớp cơ bản và gán trọng số các kết quả của mỗi bộ phân lớp. Việc lựa chọn một bộ phân lớp cơ bản sẽ ảnh hưởng tới việc lựa chọn của các bộ phân lớp cơ bản khác và trọng số được gán cho chúng.

2.2.7 Một số phương pháp kết hợp các bộ phân lớp cơ bản

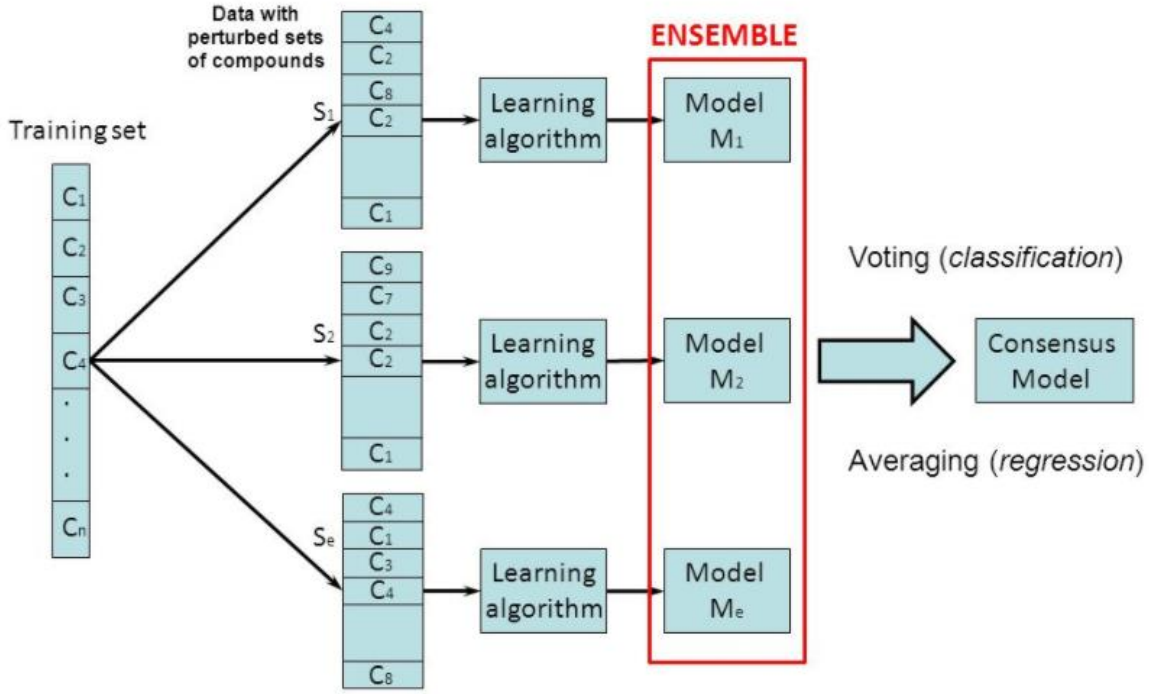
a, Phương pháp Bagging

Giới thiệu: Mô hình Bagging được Breiman đề xuất năm 1996 nhằm làm giảm lỗi variance nhưng không làm tăng lỗi bias quá nhiều.

Mô hình hoạt động: Tạo ra các bộ phân lớp từ các tập mẫu con ngẫu nhiên, chấp nhận lặp từ tập mẫu dữ liệu ban đầu, và một thuật toán học máy tương ứng. Các bộ phân lớp sẽ được kết hợp bằng phương pháp biểu quyết theo số đông. Tức là khi có một mẫu dữ liệu cần phân lớp, mỗi bộ phân lớp sẽ cho ra một kết quả. Và kết quả nào xuất hiện nhiều nhất sẽ được lấy làm kết quả của bộ kết hợp.

Thuật toán:

- Tạo ra N tập huấn luyện được chọn có lặp từ tập dữ liệu huấn luyện ban đầu. Các mẫu dữ liệu giữa các tập con huấn luyện có thể lặp nhau.
- Từ mỗi tập huấn luyện con, Bagging cho chạy với một thuật toán học máy để sinh ra tương ứng các mô hình phân lớp theo bộ phân lớp.
- Khi có một mẫu dữ liệu mới cần phân lớp, kết quả phân lớp dự đoán cuối cùng sẽ là kết quả nhận được nhiều nhất khi chạy tất cả các bộ phân lớp cơ bản thuộc tập kết hợp.



Hình 2-7: Mô hình hoạt động Bagging

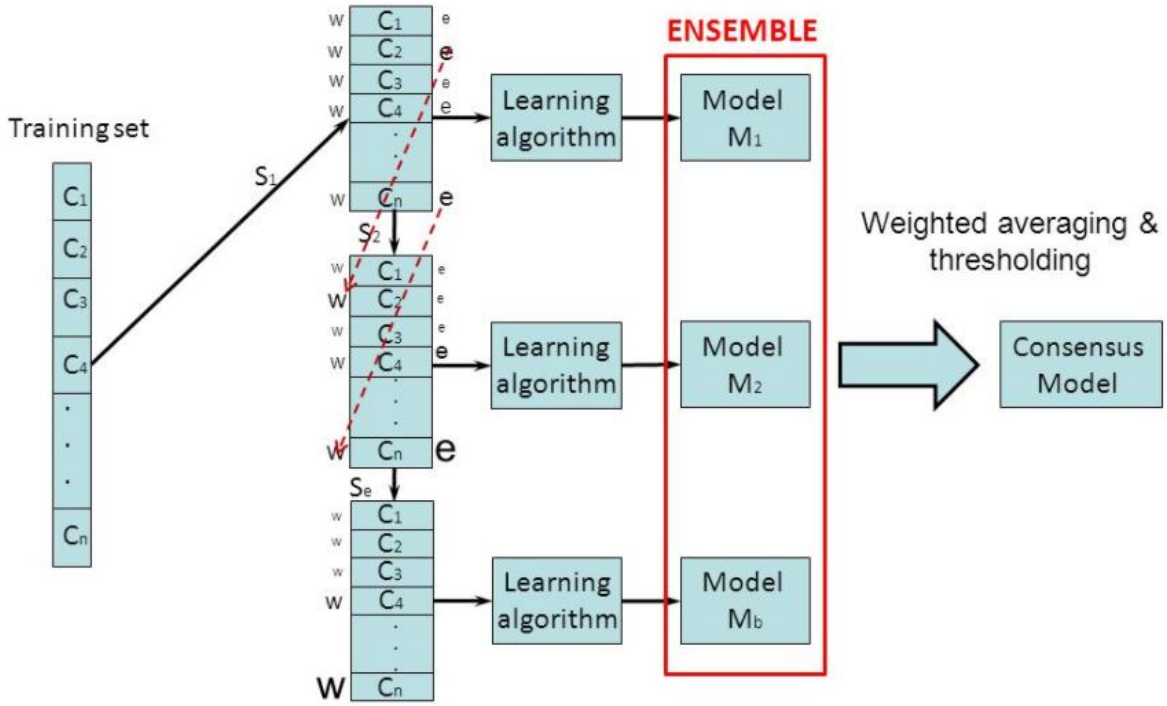
b, Phương pháp Boosting

Giới thiệu: Phương pháp *Boosting* được giới thiệu lần đầu bởi *Freund* & *Schapire* (1997), kỹ thuật này giải quyết thành công cho vấn đề phân lớp 2 lớp.

Mô hình hoạt động: Là thuật toán học quần thể bằng cách xây dựng nhiều thuật toán học cùng lúc và kết hợp chúng lại. Mục đích là để có một cụm hoặc một nhóm các bộ phân lớp yếu sau đó kết hợp chúng lại để tạo ra một phân lớp mạnh duy nhất.

Thuật toán: Ý tưởng chính của giải thuật là lặp lại quá trình học của một bộ phân lớp yếu nhiều lần. Sau mỗi bước lặp, bộ phân lớp yếu sẽ tập trung học trên các phần tử bị phân lớp sai trong các lần lặp trước. Để làm được điều này, người ta gán cho mỗi phần tử một trọng số. Khởi tạo,trọng số của các phần tử bằng nhau. Sau mỗi bước học, các trọng số này sẽ được cập nhật lại bằng cách tăng trọng số cho các phần tử bị phân lớp

sai và giảm cho các phần tử được phân lớp đúng. Kết thúc quá trình học thu được tập hợp các mô hình học dùng để phân lớp. Để phân lớp dữ liệu mới đến, người ta sử dụng luật bình chọn số đông từ kết quả phân lớp của từng mô hình phân lớp yếu.



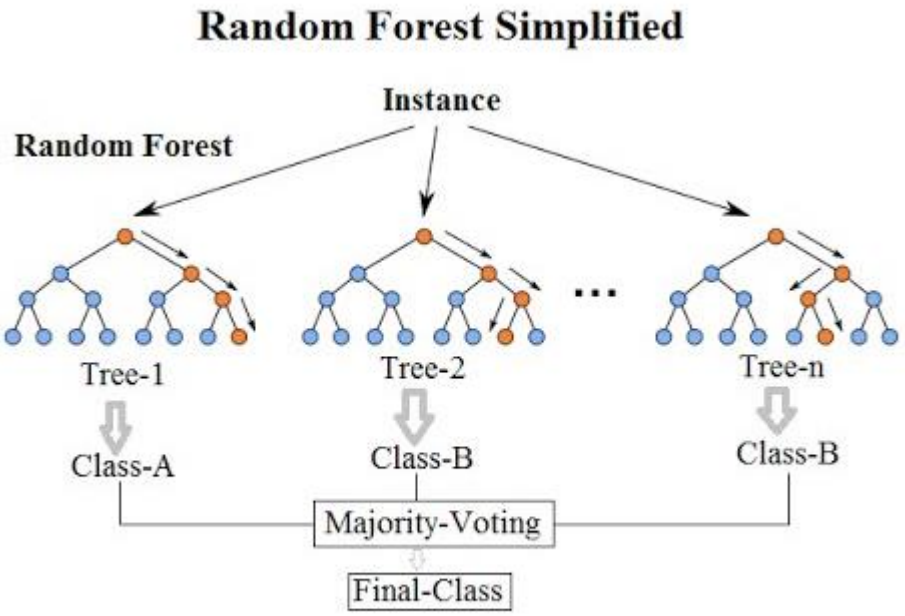
Hình 2-8: Mô hình hoạt động Boosting

c, Phương pháp Random Forest

Giới thiệu: *Random Forest* được đề xuất bởi Breiman (2001), là một trong những phương pháp tập hợp mô hình thành công nhất. Nó cho độ chính xác cao và độ chịu nhiễu tốt.

Mô hình hoạt động: Giải thuật *Random Forest* xây dựng cây không cắt nhánh nhằm giữ cho bias thấp và dùng tính ngẫu nhiên để điều khiển tính tương quan thấp giữa các cây trong rừng.

Thuật toán: *Random Forest* tạo ra một tập hợp nhiều cây quyết định không cắt nhánh, mỗi cây được xây dựng trên một tập mẫu boostrap, tại mỗi node phân hoạch tốt nhất được thực hiện từ việc chọn ngẫu nhiên một tập con các thuộc tính. Lỗi tổng quát của rừng ngẫu nhiên phụ thuộc vào độ chính xác của từng cây trong rừng và sự phụ thuộc lẫn nhau giữa các cây thành viên.



Hình 2-9: Mô hình hoạt động Random Forest

2.2.8 Đánh giá mô hình phân lớp

a, Khái niệm

Mô hình phân lớp cần được đánh giá để xem có hiệu quả không và để so sánh khả năng của các mô hình. Hiệu năng của một mô hình thường được đánh giá dựa trên tập dữ liệu kiểm định (test data). Cụ thể, giả sử đầu ra của mô hình khi đầu vào là tập dữ liệu kiểm định được mô tả bởi vector $y_{predict}$ và vector đầu ra đúng của tập kiểm định là y_{true}. Và để tính toán được hiệu năng, ta cần so sánh giữa 2 vector này với nhau.

Có nhiều cách đánh giá một mô hình. Tùy vào những bài toán khác nhau mà sử dụng cách đánh giá sao cho hợp lý. Trong phần này chúng ta tìm hiểu một số cách đánh giá cơ bản sau: accuracy, confusion matrix, true/false positive/negative…

b, Độ đo Accuracy (độ chính xác)

Cách đánh giá này tính tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm định.

Cách tính: Giả sử sau khi áp dụng mô hình phân lớp, ta thu được giá trị tham số: $Pred_{true}$ – số mẫu kiểm định dự đoán đúng, $Pred_{false}$ – số mẫu kiểm định dự đoán sai. Gọi biến *accuracy* là độ chính xác của mô hình, có giá trị theo công thức sau:

$$accuracy = \frac{Pred_{true}}{Pred_{true}+Pred_{false}} \quad (2.3)$$

c, Confusion matrix (ma trận nhầm lẫn)

Cách đánh giá Accuracy chỉ cho chúng ta biết được bao nhiêu % lượng dữ liệu được phân lớp đúng mà không chỉ ra được cụ thể mỗi loại được phân lớp như thế nào, lớp nào được phân lớp đúng nhiều nhất, và dữ liệu lớp nào thường bị phân lớp nhầm vào lớp khác. Để có thể đánh giá được các giá trị này, chúng ta sử dụng một ma trận được gọi là confusion matrix.

Bảng 2-3: Bảng giá trị ma trận confusion (chưa chuẩn hóa)

	Predict Class		
Actual Class		Positive	Negative
	Positive	TP	FN
	Negative	FP	TN

Ý nghĩa của các tham số như sau:

- TP: mẫu mang nhãn dương được phân lớp đúng vào lớp dương
- FP: mẫu mang nhãn dương bị phân lớp sai vào lớp âm
- FN: mẫu mang nhãn âm bị phân lớp sai vào lớp dương
- TN: mẫu mang nhãn âm được phân lớp đúng vào lớp âm

Gọi *accucary* là độ chính xác của mô hình sẽ được tính như sau:

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.5)$$

Cách biểu diễn ma trận như trên được gọi là unnormalized confusion matrix, nghĩa là ma trận confusion chưa chuẩn hóa. Để có ma trận confusion chuẩn hóa, ta lấy mỗi ô trên hàng của ma trận confusion chưa chuẩn hóa chia cho tổng các phần tử trên hàng đó. Như vậy, ta có nhận xét rằng tổng các phần tử trên một hàng của ma trận confusion chuẩn hóa luôn bằng 1.

Bảng 2-4: Bảng giá trị ma trận confusion (chuẩn hóa)

	Predict Class		
Actual Class		Positive	Negative
	Positive	$TP/_{TP+FN}$	$FN/_{TP+FN}$
	Negative	$FP/_{FP+TN}$	$TN/_{FP+TN}$

d, Precision & recall (độ chính xác & độ bao phủ)

Precision đối với lớp c_i:

$$Precision = \frac{TP}{TP+FP} \quad (2.6)$$

Recall đối với lớp c_i:

$$Recall = \frac{TP}{TP+FN} \quad (2.7)$$

Precision cũng được gọi là *Positive Predictive Value* và *Recall* cũng được gọi là *True Positive Rate* hay *Sensitivity* (độ nhạy).

e, Độ đo F

Độ đo *F* là một trung bình hài hòa của các tiêu chí *Precision* và *Recall*:

- *F* có xu hướng lấy giá trị gần với giá trị nào nhỏ hơn giữa hai giá trị *Precision* và *Recall*
- *F* có giá trị lớn nếu cả hai giá trị *Precision* và *Recall* đều lớn

Tiêu chí đánh giá là sự kết hợp của 2 tiêu chí đánh giá *Precision* và *Recall* theo công thức:

$$F = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (2.8)$$

CHƯƠNG 3 : DỰ ĐOÁN TƯƠNG TÁC PROTEIN - PROTEIN

Như đã đề cập ở giới thiệu mở đầu, việc nghiên cứu dự đoán tương tác protein – protein trong tin sinh học có ý nghĩa đặc biệt quan trọng trong việc tìm hiểu chức năng của protein mới, và ảnh hưởng của các hoạt động tương tác này tới tế bào trong cơ thể sống, bên cạnh các công trình nghiên cứu dự đoán tương tác PPI bằng phương pháp thực nghiệm. Đó cũng là cơ sở cho việc ra đời bài toán dự đoán tương tác protein – protein. Nội dung của bài toán trong nghiên cứu này là: đầu vào là tập các dữ liệu quan hệ tương tác giữa các cặp protein – protein đã được gán nhãn theo 2 lớp (dương tính – có tương tác, âm tính – không tương tác), qua thuật toán phân lớp tổng hợp xây dựng một mô hình để kiểm chứng đầu ra là kết quả dự đoán tương tác của các mẫu đầu vào. Từ đó suy ra độ chính xác của mô hình thuật toán.

3.1 MÔ HÌNH DỰ ĐOÁN TƯƠNG TÁC PROTEIN – PROTEIN

Để giải quyết bài toán dự đoán tương tác protein – protein. Trong những năm gần đây, rất nhiều phương pháp tin sinh học đã được đề xuất, ví dụ như: Sử dụng thông tin cấu trúc 3D của protein và tạo ra thuật toán PrePPI để dự đoán PPI ở người và nấm men [*Zhang và cộng sự,2012*] [10]. Phương pháp mở rộng mỗi polymerase: thu thập các chuỗi polypeptide ngắn liên tục xảy ra giữa các cặp tương tác protein đã biết [*Pitre & cộng sự,2006*] [11]. Sử dụng hệ thống học máy k-nearest neighbors dựa trên thành phần amino acid giả và lựa chọn thuộc tính [*Liu & cộng sự,2009*] [12]. Trích xuất thuộc tính genomic/proteomic và lựa chọn đặc trưng dự đoán PPI bằng cách sử dụng thuật toán VSM [*Urquiza & cộng sự,2011*] [13]. Sử dụng công cụ tìm kiếm cho việc truy xuất dữ liệu cơ sở dữ liệu tương tác gen để dự đoán các PPI trên cơ sở hợp nhất và hình thành gen [*Szklarczyk & cộng sự,2011*] [14].

Các phương pháp đề xuất khác nhau trong thuật toán trích xuất đặc trưng và xây dựng mô hình. Đây là 2 yếu tố ảnh hưởng đến hiệu suất của phương pháp. Đối với trích xuất thuộc tính, nhiều phương pháp khai thác thông tin đã được đề xuất. Ví dụ: Phương pháp trích xuất thuộc tính 188-D dựa vào tính chất hóa lý và sự phân bố các amino acid của protein [*Cai & cộng sự,2003*] [15], phương pháp trích xuất thuộc tính 20-D từ chuỗi protein trên cơ sở của vị trí protein – ma trận điểm riêng biệt [*Zou & cộng sự,2013*] [16], phương pháp n-gram, tạo ra từ thuật toán ngôn ngữ tự nhiên, đã được phát triển trong tin sinh học, các công cụ trích xuất đặc trưng đặc biệt như Pse-in-One, RepDNA, RepRNA … để tạo ra các thuộc tính khác nhau của chuỗi DNA, RNA và protein.

Về xây dựng mô hình, các phương pháp đề xuất có hai hướng xây dựng mô hình phân lớp: Mô hình phân lớp đơn lẻ, mô hình phân lớp tổng hợp (ensemble). Trong đó, mô hình phân lớp tổng hợp là mô hình được xây dựng từ các mô hình con của các bộ phân lớp yếu hơn, mục đích là tạo thành một bộ phân lớp mạnh. Ưu điểm của mô hình này so với các mô hình phân lớp truyền thống là có hiệu suất dự đoán tốt hơn, và lỗi dự đoán thấp hơn, nhưng có nhược điểm là có chi phí xây dựng phải bỏ ra cao hơn.

Tuy nhiên, hầu hết các phương pháp hiện tại được đề xuất đều không đề cập đến yếu tố xây dựng số liệu, đã được chứng minh là có tác động lớn đến kết quả của phương pháp tính toán. Để có được mô hình phân lớp cho dự đoán tốt nhất, dữ liệu huấn luyện cần đảm bảo vấn đề cân bằng giữa dữ liệu dương tính – âm tính. Thời điểm hiện tại, tập dữ liệu protein-protein có quan hệ tương tác - PPIs (dữ liệu dương tính), và tập dữ liệu protein – protein không có quan hệ tương tác - PPNIs (dữ liệu âm tính) đang có sự chênh lệch lớn, với số lượng PPIs lớn hơn đáng kể so với số lượng PPNIs. Ngoài ra trong tập dữ liệu âm tính PPNIs đã biết, các quan hệ không tương tác protein – protein đã được chứng minh là không có tương tác vật lý, nhưng sự không tương tác gen thì khó có thể chứng minh bằng thực nghiệm, tiềm ẩn nguy cơ tạo ra nhiều sai số giả trong số liệu âm tính PPNIs sử dụng.

Trong luận văn này, tôi nghiên cứu và xây dựng một phương pháp tính toán dự đoán tương tác protein – protein theo mô hình phân lớp tổng hợp, dựa theo các phương pháp *Bagging* của *Breiman* và cộng sự năm 1996, phương pháp *AdaBoost* của *Freund* và cộng sự năm 1997 và phương pháp *Random Forest* được phát triển bởi *Leo Breiman* và cộng sự năm 2001. Phương pháp đề xuất gồm 3 điểm chính: Xây dựng số liệu, khai thác thuộc tính, phân lớp.

- Xây dựng số liệu: sử dụng bộ số liệu dương tính, bộ số liệu âm tính đã được kiểm chứng xác thực qua các thực nghiệm sinh học.
- Khai thác thuộc tính: sử dụng 2 phương pháp là :
  - *n-gram* để xây dựng bộ thuộc tính căn cứ vào tần suất của các amino acid có mặt trong protein.
  - *Multiscale local descriptor(MLD)* chuyển chuỗi trình tự amino acid trong protein thành vector đặc trưng bằng cách sử dụng lược đồ mã hóa nhị phân.

  Sau đó áp dụng phương pháp lựa chọn thuộc tính để tạo ra một tập hợp các thuộc tính được tối ưu hóa.

- Phân lớp: sử dụng mô hình phân lớp tổng hợp, cụ thể là ba bộ phân lớp *AdaBoost*, *Bagging* và *Random Forest* vào tính toán dự đoán tương tác protein – protein và so sánh hiệu quả thu được với các bộ phân lớp đơn lẻ tương ứng được các bộ phân lớp tổng hợp sử dụng làm bộ phân lớp cơ sở, lần lượt là *Decision Stump*, *REPTree* và *Random Tree*.

Sau thực nghiệm, các kết quả cho thấy hiệu quả tốt của mô hình được xây dựng trong dự đoán PPI.

3.2 XÂY DỰNG MÔ HÌNH THỰC NGHIỆM

3.2.1 Xây dựng bộ dữ liệu

Dự đoán tương tác PPI thuộc bài toán phân lớp nhị phân, vì vậy chúng ta cần xây dựng các tập dữ liệu dương tính và âm tính. Trong đó, tập dữ liệu dương tính là tập dữ liệu chứa các cặp protein có quan hệ tương tác. Tập dữ liệu âm tính là tập dữ liệu chứa các cặp protein không có quan hệ tương tác. Trong luận văn này, tập dữ liệu dương tính được thu thập từ nguồn dữ liệu *DIP* (Database of Interacting Protein) trên Internet, có địa chỉ trang web tại: http://dip.doe-mbi.ucla.edu/dip/Main.cgi [17] với số lượng tương tác hiện có: ≈ 80.000 cặp. Tập dữ liệu âm tính ta có được từ tích lũy kết quả các thực nghiệm. Tên bộ dữ liệu âm tính là *Negatome*. Số lượng hiện có: ≈ 6.450 cặp, được lấy về từ địa chỉ trang web http://mips.helmholtz-muenchen.de/proj/ppi/negatome/ [18]

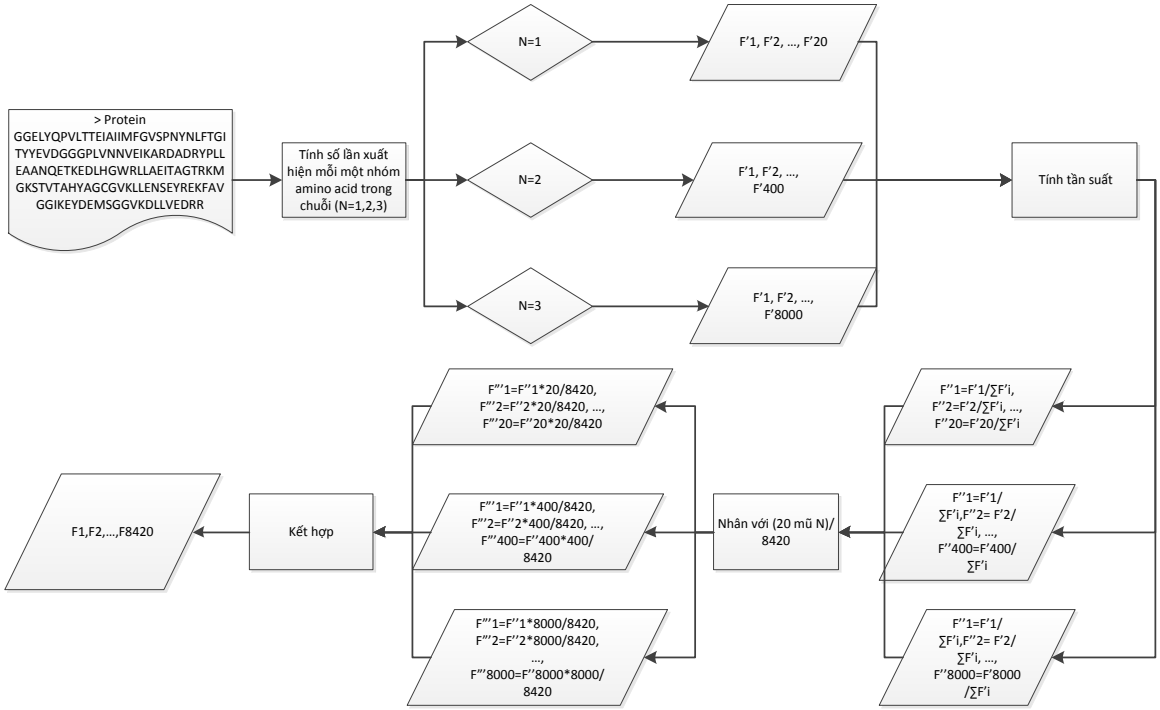
Để đảm bảo tỷ lệ dữ liệu dương tính cân bằng với dữ liệu âm tính theo tỷ lệ 1:1. Ta lấy ngẫu nhiên số lượng mẫu tập dữ liệu dương tính, tập dữ liệu âm tính *Negatome* cùng số cặp với nhau. Tổng số mẫu dữ liệu dưới dạng dữ liệu thô là chuỗi văn bản thể hiện là trình tự sắp xếp của 20 amino acid trong chuỗi protein được ký hiệu bằng chữ cái đầu của mỗi amino acid. Và để thực hiện phân lớp ta cần chuyển hóa từ dữ liệu thô sang dữ liệu dưới dạng ma trận thuộc tính dưới dạng số liệu dạng số có thể tính toán được với mỗi thuộc tính dưới dạng số là một tính chất của protein.

3.2.2 Trích xuất thuộc tính/đặc trưng

Trong nghiên cứu này tôi sử dụng 2 phương pháp trích xuất thuộc tính là *n-gram* và *Multiscale local Descriptor (MLD).*

n-gram là phương pháp được tạo ra từ thuật toán xử lý ngôn ngữ tự nhiên [19]. Các *n-gram* được sử dụng để mã hóa protein được xây dựng bằng cách tính tần số xuất hiện của n chuỗi amino acid. Với tần số tính bằng tổng các thuộc tính hoặc tổng số lần xuất hiện của mỗi thuộc tính. Phương pháp tần suất *n–gram* có thể được sử dụng để đạt các

thuộc tính *1-gram*, *2-gram* và *3-gram*. Vì 3 loại thuộc tính đều có đóng góp khác nhau cho quan hệ tương tác, nên để đầy đủ, tôi đã thực hiện nhân ba thuộc tính theo các trọng lượng khác nhau để tạo ra một vector đặc trưng có 8420 chiều.



Hình 3-1: Sơ đồ phương pháp trích xuất thuộc tính *n-gram*

Vì các cặp PPI, PPNI bao gồm 2 protein nên khi trích xuất thuộc tính bởi phương pháp n-gram sẽ sinh ra 2 chuỗi mã. Để sử dụng cần kết hợp 2 vector thuộc tính riêng để tạo ra vector thuộc tính cuối cùng. Giả định rằng PPI (hoặc PPNI) bao gồm 2 protein A và B. Protein A, B được mã hóa bởi phương pháp trích xuất đặc trưng được biểu diễn bằng FA và FB, trong đó FA và FB là 2 vector thuộc tính biểu diễn bởi:

$$FA = \{FA_1, FA_2, FA_3, \dots, FA_{8420}\}$$

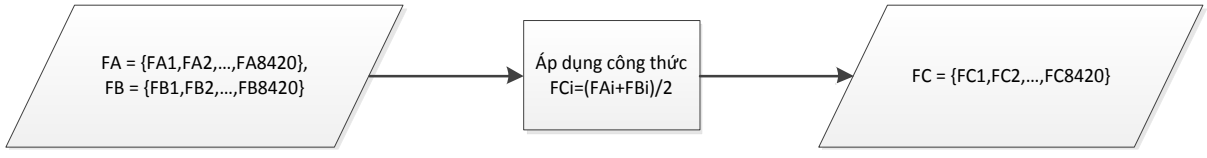
$$FB = \{FB_1, FB_2, FB_3, \dots, FB_{8420}\}$$

Gọi FC là vector thuộc tính kết hợp. Ta có:

$$FC = \{FC_1, FC_2, FC_3, \dots, FC_{8420}\}$$

với:

$$FC_i = \frac{FA_i + FB_i}{2}$$



Hình 3-2: Sơ đồ kết hợp 2 vector thuộc tính của cặp protein - protein

MLD là phương pháp được đề xuất để biến đổi chuỗi trình tự amino acid trong protein thành các vector đặc trưng bằng cách sử dụng một lược đồ mã hóa nhị phân. Toàn bộ chuỗi trình tự amino acid trong protein được chia thành 4 đoạn có độ dài bằng nhau, với số lượng amino acid được giảm chiều bằng cách chia 20 amino acid thành 7 nhóm cơ bản dựa vào tính lưỡng cực và thể tích mạch nhánh của amino acid, giả sử ký hiệu 4 đoạn theo thứ tự là: S1, S2, S3, S4. Xét tính liên tục của 4 đoạn với nhau chia chuỗi trình tự ban đầu thành 9 chuỗi trình tự con, biểu diễn dưới dạng mã hóa nhị phân 4 bit là: 1000, 0100, 0010, 0001, 1100, 0110, 0011, 1110, 0111, trong đó ký tự 1 biểu diễn đoạn tương ứng có tồn tại, ký tự 0 biểu diễn đoạn tương ứng không tồn tại. Với mỗi chuỗi trình tự con, ta tính toán các mô tả địa phương: Thành phần, chuyển tiếp và phân bố. Thành phần tính tần suất của mỗi nhóm trên tổng số phần tử; Chuyển tiếp tính tần suất của các amino acid trong một nhóm có phần tử kế tiếp là amino acid thuộc một nhóm khác; Phân bố xác định tần suất ở các vị trí đầu tiên, vị trí 25%, 50%, 75% và vị trí cuối cùng của nhóm trong chuỗi trình tự con. Cuối cùng, mỗi một chuỗi trình tự con có 63 mô tả được tạo ra: 7 mô tả thành phần, 21 mô tả chuyển tiếp, 35 mô tả phân bố. Mỗi protein chia thành 9 chuỗi trình tự con có các mô tả ghép lại tạo ra một vector $63 * 9 = 567$ chiều. Cặp protein PPI (hoặc PPNI) được kết hợp để tạo ra vector đặc trưng cuối cùng bằng cách ghép 2 vector 567 chiều của mỗi protein, sinh ra một vector 1134 chiều đại diện cho cặp protein đó [20].

Bảng 3-1: Bảng chia nhóm 20 amino acid dựa vào tính lưỡng cực và khối lượng mạch nhánh

Nhóm	*Amino acid*	*Tính lưỡng cực*	*Khối lượng mạch nhánh*
1	A, G, V	dipole < 1	volume < 50
2	C	1 < dipole < 2	volume > 50
3	M, S, T, Y	1 < dipole < 2	volume > 50
4	F, I, L, P	dipole < 1	volume > 50
5	H, N, Q, W	2 < dipole < 3	volume > 50
6	K, R	dipole > 3	volume > 50
7	D, E	dipole > 3	volume > 50

Sau bước trích xuất thuộc tính ta có một ma trận thuộc tính kích thước $(m \times n)$:

$$\begin{pmatrix} A_{00} & A_{01} \dots & A_{0(n-1)} & 1 \\ \dots & \dots \dots & \dots & \dots \\ A_{m0} & A_{m1} \dots & A_{m(n-1)} & 0 \end{pmatrix}$$

với m là số lượng bộ dữ liệu, n là số thuộc tính của bộ dữ liệu, bao gồm cả thuộc tính phân lớp. Trong nghiên cứu này, thuộc tính phân lớp quy ước có 2 giá trị: giá trị 1 – tương ứng lớp định nghĩa có quan hệ tương tác protein – protein, giá trị 0 – tương ứng lớp định nghĩa không có quan hệ tương tác protein – protein.

3.2.3 Lựa chọn thuộc tính/đặc trưng

Không phải tất cả các thuộc tính trích xuất được đều có lợi cho việc phân lớp. Vì vậy, lựa chọn các thuộc tính có độ quan trọng cao trong bộ dữ liệu thuộc tính ban đầu là cần thiết trước khi áp dụng các giải thuật phân lớp thuộc tính. Trong luận văn này, phương pháp MRMD được sử dụng. Mục tiêu chính của phương pháp là tìm kiếm một loại chỉ số xếp hạng của thuộc tính đáp ứng 2 yêu cầu, đó là: sự liên quan giữa tập hợp thuộc tính và lớp đích, và tính thừa của bộ thuộc tính. Hệ số tương quan Pearson được sử dụng để đo lường sự liên quan. Ba loại hàm khoảng cách (ED, khoảng cách Cosine, và hệ số Tanimoto) được sử dụng để tính toán sự thừa. Sự liên quan giữa tập thuộc tính và lớp đích tăng lên cùng với sự gia tăng hệ số tương quan của Pearson. Khoảng cách giữa các thuộc tính càng lớn thì độ thừa của tập thuộc tính càng thấp. Thuộc tính với tổng lớn hơn của sự liên quan và khoảng cách được chọn làm bộ thuộc tính cuối cùng. Kết quả, bộ thuộc tính do MRMD tạo ra có sự dư thừa thấp và độ liên quan cao tới lớp đích.

Cụ thể, ở đây ta tính hệ số tương quan Pearson giữa lớp mỗi thuộc tính trong dữ liệu đầu vào và lớp đích là lớp nhãn phân lớp theo công thức sau:

$$r_t = \frac{numerator}{denominator} \tag{3.1}$$

$$numerator = \sum_{i=1}^{n}(x_i - \bar{x}) * (y_i - \bar{y}) \tag{3.2}$$

$$denominator = \sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.3}$$

Trong đó:

x_i: giá trị của cột thuộc tính X trên đối tượng i

$\bar{x}$: kỳ vọng của cột thuộc tính X

y_i: giá trị của cột lớp nhãn Y trên đối tượng i

$\bar{y}$: kỳ vọng của cột lớp nhãn Y

n: số đối tượng trong tập đầu vào

r_i: hệ số tương quan Pearson giữa cột thuộc tính t và cột lớp nhãn Y

Ba loại hàm khoảng cách được tính như sau:

Độ đo Euclid của thuộc tính X được tính theo công thức:

$$ED_X = \frac{\sum_{i=1}^{k} ED_{XY_i}}{k} \quad (3.4)$$

với k là số thuộc tính và ED_{XY} là độ đo EuClid giữa hai thuộc tính X và Y được tính theo công thức:

$$ED_{XY} = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2} \quad (3.5)$$

Trong đó:

n: số đối tượng thuộc tập đầu vào

x_i: giá trị thuộc tính X của đối tượng thứ i, hay $x_i \in (X = \{x_1, x_2, \dots, x_n\})$

y_i: giá trị thuộc tính Y của đối tượng thứ i, hay $y_i \in (Y = \{y_1, y_2, \dots, y_n\})$

Độ đo Cosine của thuộc tính X được tính theo công thức:

$$Cosine_X = \frac{\sum_{i=1}^{k} Cosine_{XY_i}}{k} \quad (3.6)$$

k là số thuộc tính và $Cosine_{XY}$ là giá trị độ đo Cosine giữa 2 thuộc tính X và Y được tính bởi công thức:

$$Cosine_{XY} = \frac{\sum_{i=1}^{n} x_i * y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} * \sqrt{\sum_{i=1}^{n} y_i^2}} \quad (3.7)$$

Trong đó:

n: số đối tượng thuộc tập đầu vào

x_i: giá trị thuộc tính X của đối tượng thứ i, hay $x_i \in (X = \{x_1, x_2, \dots, x_n\})$

y_i: giá trị thuộc tính Y của đối tượng thứ i, hay $y_i \in (Y = \{y_1, y_2, \dots, y_n\})$

Độ đo Tanimoto của thuộc tính X được tính theo công thức:

$$Tanimoto_X = \frac{\sum_{i=1}^{k} Tanimoto_{XY_i}}{k} \quad (3.8)$$

với k là số thuộc tính và $Tanimoto_{XY}$ là giá trị độ đo Tanimoto giữa 2 thuộc tính X và Y được tính bằng công thức:

$$Tanimoto_{XY} = \frac{\sum_{i=1}^{n} x_i * y_i}{\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n} x_i * y_i} \quad (3.9)$$

Trong đó:

n: số đối tượng thuộc tập đầu vào

x_i: giá trị thuộc tính X của đối tượng thứ i, hay $x_i \in (X = \{x_1, x_2, \dots, x_n\})$

y_i: giá trị thuộc tính Y của đối tượng thứ i, hay $y_i \in (Y = \{y_1, y_2, \dots, y_n\})$

Xếp hạng độ liên quan cao và độ dư thừa thấp của các thuộc tính theo phương pháp MRMD được tính theo công thức:

$$mrmd_i = \frac{(r_i+ED_i)+(r_i+Cosine_i)+(r_i+Tanimoto_i)}{3} \quad (3.10)$$

Sau đó, thực hiện lấy các thuộc tính có giá trị xếp hạng cao nhất vào tập thuộc tính được lựa chọn.

Sau bước này, ta lựa chọn được tập thuộc tính quan trọng nhất từ tập thuộc tính ban đầu thỏa mãn điều kiện có độ liên quan cao tới lớp thuộc tính phân lớp và có độ dư thừa thấp trong tập các thuộc tính. Đây là bộ dữ liệu dùng làm đầu vào cho việc phân lớp và đánh giá kết quả phân lớp. Ta sử dụng phương pháp *k-fold cross validation*, trong nghiên cứu này sử dụng k = 10, hay chia tập thuộc tính đặc trưng thành 10 phần bằng nhau, 9 phần sử dụng làm dữ liệu huấn luyện, và phần còn lại là dữ liệu kiểm định mô hình.

Để đa dạng tập dữ liệu huấn luyện và tập dữ liệu kiểm định, ta xây dựng một hàm chia file và đảm bảo xáo trộn dữ liệu trong tập ban đầu trước khi chia.

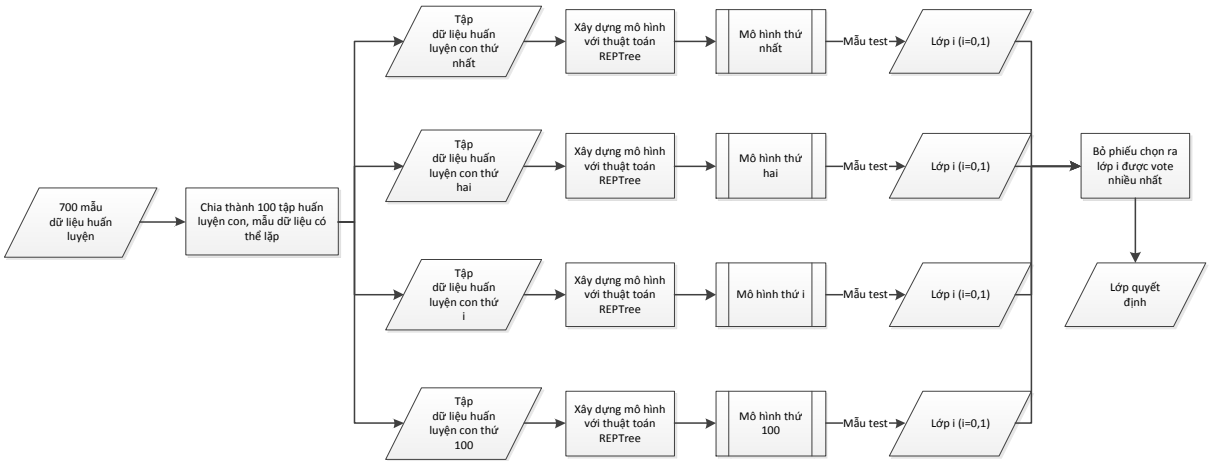
3.2.4 Phân lớp đặc trưng

Trong nghiên cứu này, ta thực nghiệm xử lý phân lớp theo hướng sử dụng thuật toán phân lớp tổng hợp với ba bộ phân lớp là: *Bagging*, *AdaBoostM1* và *Random Forest* để làm rõ ưu điểm so với các thuật toán phân lớp đơn lẻ sử dụng đối chứng trong nghiên cứu là *Decision Stump*, *REPTree* và *Random Tree*.

Giả sử tập dữ liệu thuộc tính đặc trưng thu được có số lượng *n* mẫu, ta phân chia làm 10 phần bằng nhau. Với 10 phần ta chia làm 10 bộ dữ liệu huấn luyện và dữ liệu test theo cách: lấy một phần làm dữ liệu test thì 9 phần còn lại là dữ liệu huấn luyện, lặp lại cho 10 phần dữ liệu đều sử dụng làm dữ liệu test. Tập dữ liệu huấn luyện có số lượng $n_1 = n \times 0{,}9$ mẫu, tập dữ liệu kiểm định có số lượng $n_2 = n \times 0{,}1$ mẫu. Trong đó 2 tập dữ liệu huấn luyện và tập dữ liệu kiểm định độc lập với nhau và không được có phần tử chung, đảm bảo việc kiểm định là khách quan nhất.

Trong đó bộ phân lớp tổng hợp *Bagging* sử dụng thuật toán cơ bản là *REPTree*, với dữ liệu huấn luyện là n_1 mẫu huấn luyện. Từ n_1 mẫu huấn luyện ta tạo ra k tập dữ liệu huấn luyện con, trong đó các mẫu huấn luyện được chọn ngẫu nhiên và có thể có lặp. Tạo tương ứng các mô hình với mỗi tập huấn luyện trong k tập huấn luyện con cùng thuật

toán *REPTree*, ta thu được k mô hình cơ bản trong *Bagging*. Với mỗi mẫu cần dự đoán mới trong n_2 mẫu dữ liệu kiểm định đi vào trong *Bagging*, ta thực hiện dự đoán phân lớp mẫu này qua k mô hình cơ bản và biểu quyết mẫu này thuộc lớp nào có số lượng bỏ phiếu nhiều nhất.



Hình 3-3: Sơ đồ thuật toán Bagging trên tập n_1 mẫu huấn luyện

Thứ hai, bộ phân lớp tổng hợp *AdaBoostM1* trong nghiên cứu này sử dụng thuật toán cơ bản là *Decision Stump* (cây quyết định một cấp). Cách thực hiện giải thuật *AdaBoostM1* là thực hiện xây dựng lặp lại các mô hình cơ bản trên tập dữ liệu huấn luyện có trọng số thay đổi sau mỗi lần traning, theo hướng: ở vòng training trước, mẫu dữ liệu nào dự đoán đúng sẽ gán trọng số thấp đi, mẫu dữ liệu nào dự đoán sai sẽ được gán trọng số cao hơn, mục đích là ở vòng training sau mẫu dữ liệu sai này sẽ có vai trò quan trọng hơn trong việc phân lớp. Chương trình sẽ thoát khi ta có tỉ lệ mẫu dự đoán sai >=1/2 (tổng số mẫu), hoặc kết thúc số lần training cài đặt.

Cuối cùng, *Random Forest* sử dụng thuật toán cơ bản là *Random Tree* (cây ngẫu nhiên). Cách thực hiện giải thuật như sau:

1. Xây dựng *Random Tree* đầu tiên trong *Random Forest*:
   1.1. Lựa chọn k mẫu ngẫu nhiên trong tổng số n_1 mẫu tập học làm tập dữ liệu học.
   1.2. Lựa chọn gốc là thuộc tính có giá trị phân lớp tập đích tốt nhất trong tất cả các thuộc tính. Trong nghiên cứu này, số thuộc tính lựa chọn mặc định bằng số thuộc tính trong tập học.
   1.3. Tính các node trong tiếp theo sau gốc bằng cách chọn thuộc tính có giá trị phân chia tốt nhất trong các thuộc tính còn lại.

1.4. Chỉ dừng cho đến khi thu được một cây hoàn chỉnh có gốc và tận cùng là các nút lá với chiều sâu cây mong muốn. Trong nghiên cứu này, chiều sâu cây không bị giới hạn cho thuộc tính cuối cùng tách được.

2. Thuật toán *Random Forest* tạo k *Random Tree* tương ứng hàm xây dựng bước 1 được lặp lại k lần.
3. Thực hiện dự đoán dữ liệu kiểm định bởi mỗi *Random Tree* vừa xây dựng, tổng hợp k kết quả dự đoán từ k *Random Tree*, tính toán số lượng lớp được dự đoán. Kết quả dự đoán cuối cùng của thuật toán *Random Forest* là thuộc lớp có số lượng được dự đoán nhiều hơn.

CHƯƠNG 4 KẾT QUẢ THỰC NGHIỆM VÀ KẾT LUẬN

4.1 CHƯƠNG TRÌNH CÀI ĐẶT

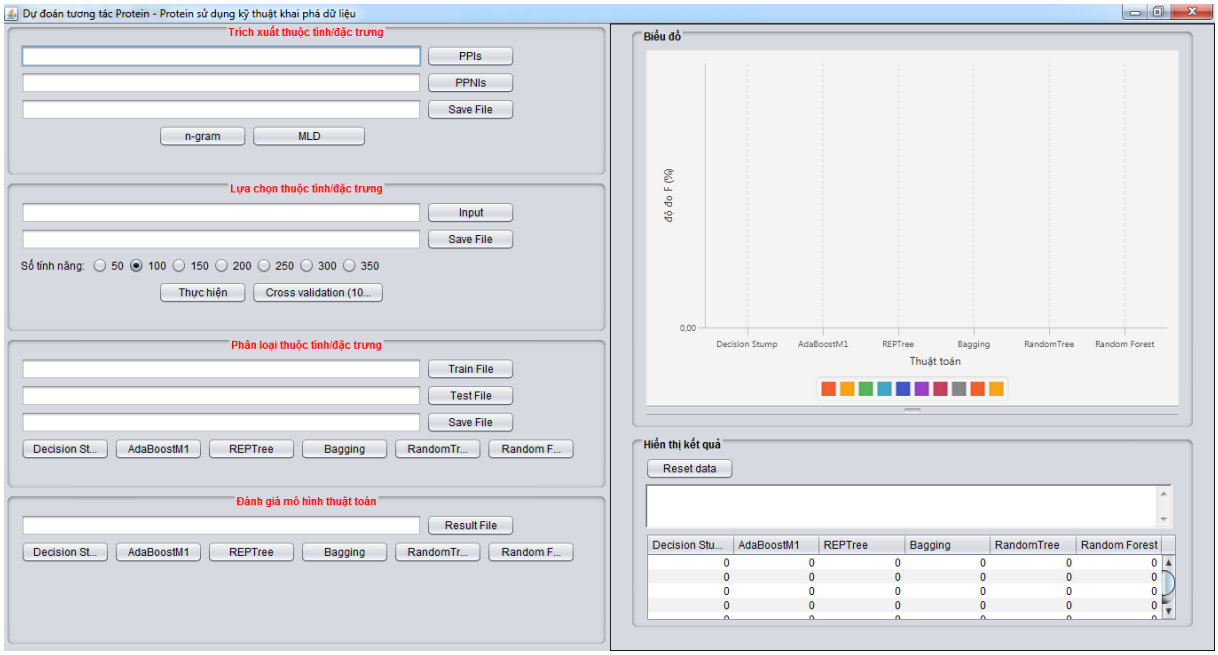
4.1.1 Yêu cầu cấu hình

Chương trình thực nghiệm dự đoán tương tác protein - protein sử dụng kỹ thuật khai phá dữ liệu được lập trình bằng ngôn ngữ Java. Yêu cầu cần có để chạy được chương trình là:

- Môi trường java tối thiểu version 1.6
- Phần cứng:
    - CPU Dual-core+, RAM 8G+ (cho trường hợp chạy lựa chọn thuộc tính/đặc trưng sau trích xuất thuộc tính/đặc trưng *n-gram*)
    - CPU Dual-core+, RAM 4G+ (cho trường hợp chạy lựa chọn thuộc tính/đặc trưng sau trích xuất thuộc tính/đặc trưng *MLD*)
- Client chạy ứng dụng phải là máy cài hệ điều hành Windows.

4.1.2 Cài đặt

Mở giao diện chương trình:



Hình 4-1: Giao diện chương trình Dự đoán tương tác protein – protein sử dụng kỹ thuật khai phá dữ liệu

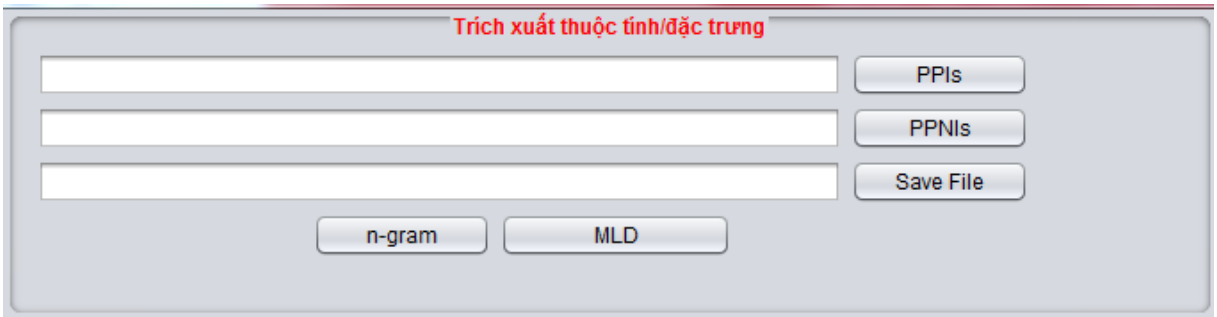
a, Chuẩn bị dữ liệu

Dữ liệu dương tính: Tải về từ nguồn DIP có địa chỉ tại: http://dip.doe-mbi.ucla.edu/dip/Main.cgi. Số lượng các cặp PPI lấy ngẫu nhiên 6445 cặp.

Dữ liệu âm tính: Tải về từ nguồn có địa chỉ tại http://mips.helmholtz-muenchen.de/proj/ppi/negatome/. Số lượng PPNI lấy ngẫu nhiên: 6445 cặp.

Dữ liệu có dạng tệp nén chứa các file đuôi *.fasta, trong mỗi file có dữ liệu thô chứa thông tin về cặp protein.

b, Trích xuất thuộc tính/đặc trưng



Hình 4-2: Giao diện chức năng trích xuất thuộc tính/đặc trưng

Nhấn button [PPIs], chọn thư mục chứa các cặp protein tương tác.

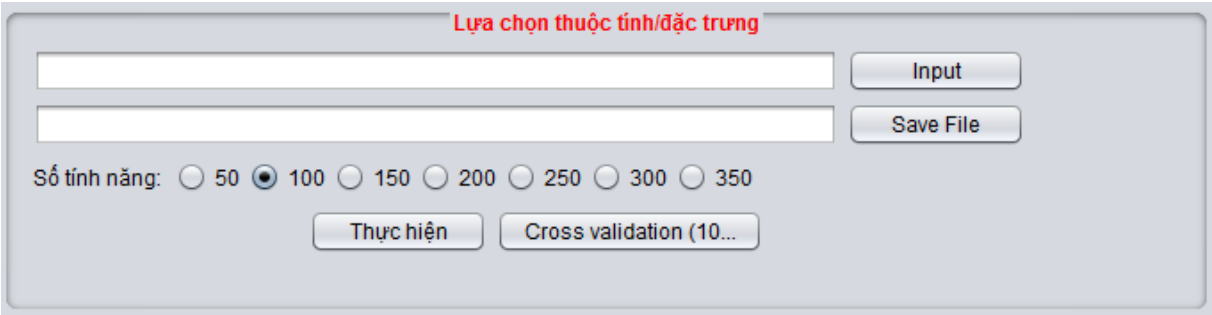
Nhấn button [PPNIs], chọn thư mục chứa các cặp protein không tương tác.

Nhấn button [Save File], chọn thư mục lưu file kết quả trích xuất.

Nhấn button [n-gram] - thực hiện trích xuất thuộc tính/đặc trưng theo phương pháp *n-gram*

Nhấn button [MLD] – thực hiện trích xuất thuộc tính/đặc trưng theo phương pháp *MLD*.

c, Lựa chọn thuộc tính/đặc trưng



Hình 4-3: Giao diện chức năng lựa chọn thuộc tính/đặc trưng

Nhấn [Input] chọn file dữ liệu trích xuất được ở bước *b, Trích xuất thuộc tính/đặc trưng* làm đầu vào.

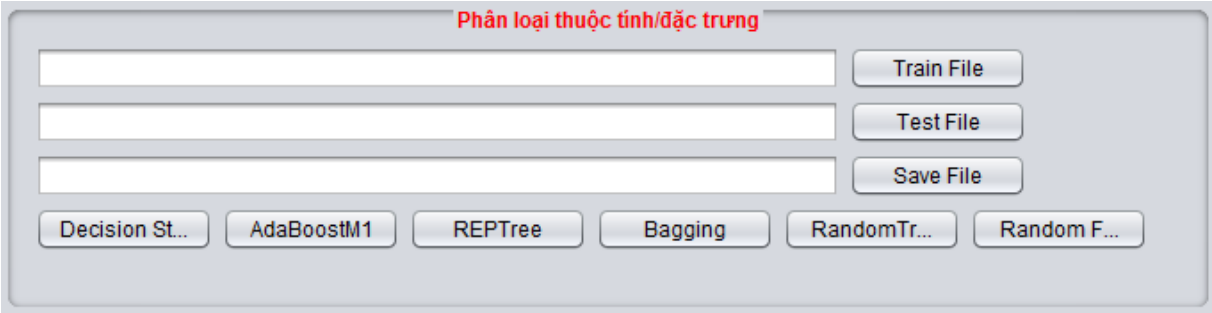
Nhấn [Save File] chọn thư mục cần lưu file kết quả lựa chọn thuộc tính/đặc trưng.

Nhấn [Thực hiện] thực hiện gọi hàm lựa chọn tính năng/đặc trưng.

Nhấn [Cross validation (10-fold)] thực hiện chia file kết quả sau khi lựa chọn thuộc tính/đặc trưng thành 10 phần dữ liệu bằng nhau, sử dụng lần lượt mỗi phần dữ liệu làm

dữ liệu kiểm định, 9 phần còn lại làm dữ liệu training. Ta có 10 bộ dữ liệu, mỗi bộ dữ liệu gồm 2 file: file dữ liệu training và file dữ liệu kiểm định.

d, Phân lớp đặc trưng



Hình 4-4: Giao diện chức năng Phân lớp thuộc tính/đặc trưng

Nhấn button [Train File] chọn file dữ liệu huấn luyện.

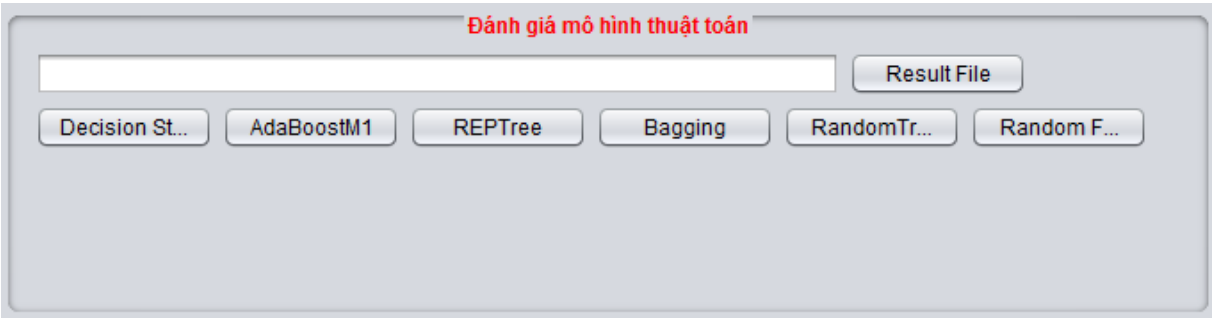
Nhấn button [Test File] chọn file dữ liệu kiểm định.

Nhấn button [Save File] chọn thư mục lưu file kết quả phân lớp từ đầu vào là dữ liệu kiểm định.

Nhấn 1 trong 6 button [Decision Stump], [AdaBoostM1], [REPTree], [Bagging], [Random Tree] hoặc [Random Forest] thực hiện gọi hàm phân lớp đặc trưng tương ứng cho mỗi thuật toán: thuật toán phân lớp đơn lẻ *Decision Stump*, *REPTree* hoặc *Random Tree*, thuật toán phân lớp tổng hợp *AdaBoostM1*, *Bagging*, hoặc *Random Forest*.

Kết quả dự đoán và biểu đồ tương ứng hiển thị trong hộp Panel bên phải.

e, Độ đo đánh giá



Hình 4-5: Giao diện chức năng Đánh giá mô hình thuật toán

Nhấn button [Result File] chọn file kết quả vừa thu được qua bước phân lớp thuộc tính/đặc trưng.

Nhấn 1 trong 6 button [Decision Stump], [AdaBoostM1], [REPTree], [Bagging], [Random Tree] hoặc [Random Forest] để thực hiện gọi hàm tính toán độ đo tương ứng cho mỗi thuật toán phân lớp *Decision Stump*, *AdaBoostM1*, *REPTree*, *Bagging*, *Random Tree* hoặc *Random Forest*.

Kết quả đánh giá độ đo và biểu đồ tương ứng hiển thị trong hộp Panel bên phải.

4.2 KẾT QUẢ DỰ ĐOÁN TƯƠNG TÁC PROTEIN - PROTEIN

Tiến hành thực nghiệm với 6 thuật toán, 3 thuật toán phân lớp tổng hợp là *AdaBoostM1*, *Bagging* và *Random Forest*, 3 thuật toán phân lớp đơn lẻ là *Decision Stump*, *REPTree*, và *RandomTree*. Như đã đề cập ở phần 3.3.3. Lựa chọn thuộc tính/đặc trưng, ta áp dụng phương pháp *k-fold cross validation*, bằng cách xây dựng một hàm chia file dữ liệu ban đầu thành 10 phần bằng nhau. Lấy lần lượt mỗi phần làm dữ liệu kiểm định và 9 phần còn lại làm dữ liệu huấn luyện, ta thu được 10 bộ dữ liệu. Mỗi bộ dữ liệu có 2 file: file dữ liệu huấn luyện và file dữ liệu kiểm định với tỉ lệ 9:1. Để kết quả thu được có sự khách quan, trong nghiên cứu này, trước khi chia ta xáo trộn dữ liệu trong file ban đầu một cách ngẫu nhiên nếu người dùng muốn có nhiều hơn các bộ dữ liệu đánh giá.

Sau khi chạy chương trình từ các bộ dữ liệu huấn luyện và bộ dữ liệu kiểm định vừa chia, ta thu được các file kết quả với thuật toán tương ứng, với mỗi mẫu dữ liệu trong tập mẫu kiểm định trong file có giá trị thuộc 2 cột: Cột lớp đích được dự đoán, và cột lớp đích đúng cho trước.

Bảng 4-1: Bảng giá trị phân lớp dự đoán

Mẫu dữ liệu	Lớp dự đoán	Lớp đúng
1	0	1
2	1	1
3	1	1
4	0	0
5	1	1
6	0	1
…	…	…
1285	1	1
1286	0	0
1287	1	1
1288	1	0
1289	0	0

Để biểu diễn kết quả ngắn gọn và tường minh, trong nghiên cứu sử dụng độ đo F để hiển thị trên chương trình tương ứng với các thuật toán trên mỗi bộ dữ liệu. Ta hiển thị kết quả theo 2 hướng: sử dụng thuật toán trích xuất thuộc tính/đặc trưng *n-gram* và thuật toán trích xuất thuộc tính/đặc trưng *MLD*.

Sau bước trích xuất thuộc tính/đặc trưng, ta lựa chọn thuộc tính với số thuộc tính lựa chọn nhỏ hơn số thuộc tính ban đầu. Trong nghiên cứu này, ta thực hiện lựa chọn thuộc

tính với số thuộc tính rút gọn là 100 thuộc tính và so sánh kết quả phân lớp đặc trưng của tập dữ liệu ban đầu và tập dữ liệu đã rút gọn thuộc tính.

Sử dụng thuật toán trích xuất thuộc tính/đặc trưng *MLD*, sau đó ta nghiên cứu kết quả theo 2 hướng: Hướng thứ nhất, dùng nguyên tập thuộc tính ban đầu làm đầu vào cho các thuật toán phân lớp, và hướng thứ hai, dùng phương pháp MRMD giảm bớt số chiều thuộc tính từ 1134 thuộc tính xuống còn 100 thuộc tính.

Trong bảng biểu diễn kết quả ta xếp thành từng cặp theo luật: thuật toán phân lớp đơn lẻ A - thuật toán phân lớp tổng hợp có cơ sở là thuật toán phân lớp đơn lẻ A tương ứng. Cụ thể là các cặp: *Decision Stump - AdaBoostM1*, *REPTree – Bagging*, *RandomTree – RandomForest*.

Ta có kết quả thực nghiệm đo được với phương pháp trích xuất thuộc tính/đặc trưng *MLD,* giữ nguyên tập 1134 thuộc tính sau trích xuất làm tập đầu vào phân lớp như sau:

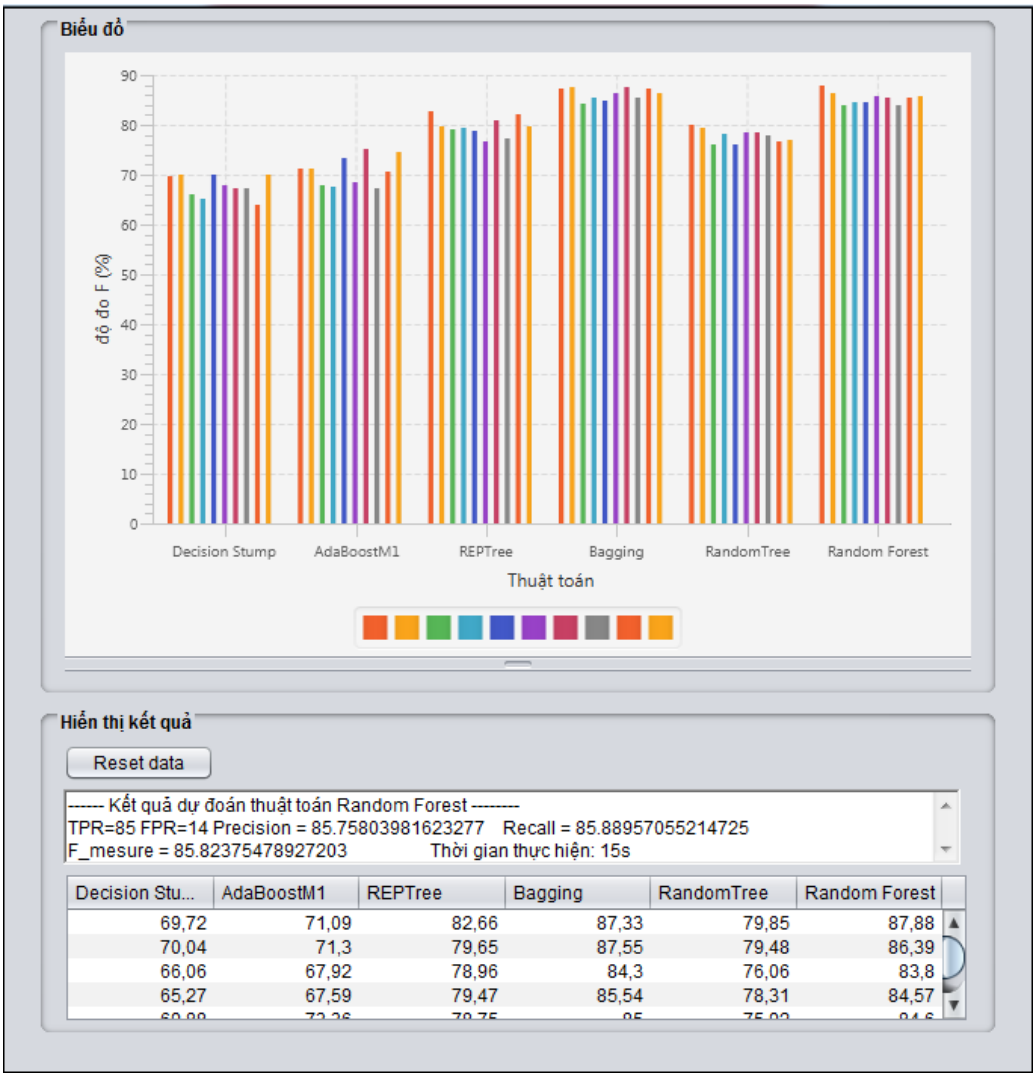
Bảng 4-2: Kết quả thực nghiệm phương pháp trích xuất thuộc tính MLD, không giảm chiều số thuộc tính

Độ đo F (%)	Decision Stump	AdaBoost	REPTree	Bagging	Random Tree	Random Forest
S1	69,72	71,09	82,66	87,33	79,85	87,88
S2	70,04	71,30	79,65	87,55	79,48	86,39
S3	66,06	67,92	78,96	84,3	76,06	83,80
S4	65,27	67,59	79,47	85,54	78,31	84,57
S5	69,88	73,36	78,75	85	75,92	84,60
S6	68,03	68,45	76,74	86,25	78,50	85,76
S7	67,41	75,17	81,05	87,62	78,54	85,43
S8	67,28	67,21	77,26	85,58	77,91	84,09
S9	64,01	70,46	82,15	87,42	76,65	85,56
S10	69,90	74,55	79,60	86,33	76,92	85,82
Trung bình	**67,76**	**70,71**	**79,63**	**86,29**	**77,81**	**85,39**

Trong đó, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10 là các bộ dữ liệu kiểm định sử dụng trong thực nghiệm. Kết quả được biểu diễn bằng giá trị độ đo F (%).

Bảng 4-3: Thời gian thực hiện phương pháp trích xuất thuộc tính MLD, không giảm chiều số thuộc tính

Thời gian (s)	Decision Stump	AdaBoost	REPTree	Bagging	Random Tree	Random Forest
MLD – không giảm chiều thuộc tính	18;21;22; 21;17;18; 18;17;17; 17	107;129; 140;95; 94;101; 97;98; 102;105	37;38;40; 33;34;32; 32;34;35; 34	225;222; 235;230; 224;220; 220;224; 226;230	9;9;10;9; 9;9;9;9; 9;9	16;16;17; 16;15;15; 15;15;16; 15
	18,6	**106,8**	**34,9**	**225,6**	**9,1**	**15,6**



Hình 4-6: Biểu đồ kết quả thực nghiệm phương pháp trích xuất thuộc tính MLD, không giảm chiều số thuộc tính

Sau khi trích xuất thuộc tính/đặc trưng MLD, giảm chiều thuộc tính xuống còn 100 thuộc tính, ta có kết quả thực nghiệm đo được như sau:

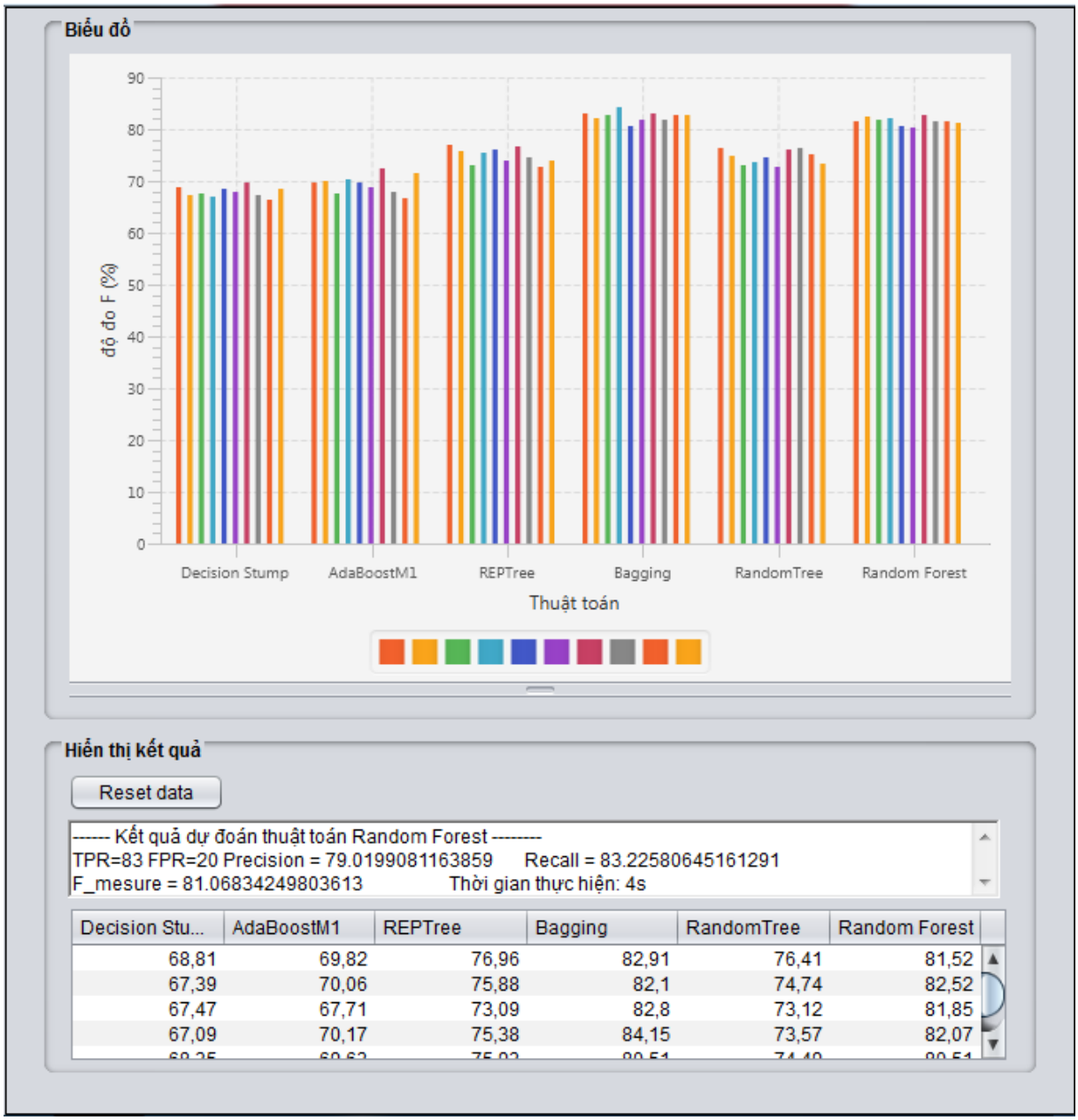
Bảng 4-4: Kết quả thực nghiệm phương pháp trích xuất thuộc tính MLD, giảm chiều còn 100 thuộc tính

Độ đo F (%)	Decision Stump	AdaBoost	REPTree	Bagging	Random Tree	Random Forest
S1	68,81	69,82	76,96	82,91	76,41	81,52
S2	67,39	70,06	75,88	82,10	74,74	82,52
S3	67,47	67,71	73,09	82,80	73,12	81,85
S4	67,09	70,17	75,38	84,15	73,57	82,07
S5	68,35	69,63	75,93	80,51	74,49	80,51
S6	67,90	68,84	73,88	81,72	72,66	80,28
S7	69,69	72,42	76,59	82,94	76,17	82,81
S8	67,13	67,80	74,68	81,80	76,39	81,66
S9	66,42	66,67	72,86	82,64	75,04	81,46
S10	68,46	71,45	73,92	82,80	73,36	81,07
Trung bình	**67,87**	**69,46**	**74,92**	**82,44**	**74,60**	**81,58**

Trong đó, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10 là các bộ dữ liệu kiểm định sử dụng cho thực nghiệm. Kết quả biểu diễn bằng giá trị độ đo F (%).

Bảng 4-5: Thời gian thực hiện phương pháp trích xuất thuộc tính MLD, giảm chiều còn 100 thuộc tính

Thời gian (s)	Decision Stump	AdaBoost	REPTree	Bagging	Random Tree	Random Forest
MLD – 100 thuộc tính	3;4;4;3;2;2;1;1;1;1	6;9;8;8;6;8;7;10;6;7	3;2;2;2;2;2;2;2;2;2	18;18;17;17;18;17;18;17;19;18	1;1;1;1;1;1;1;1;1;1	4;4;4;4;4;4;4;4;4;4
	2,2	**7,5**	**2,1**	**17,7**	**1**	**4**



Hình 4-7: Biểu đồ kết quả thực nghiệm phương pháp trích xuất thuộc tính MLD, giảm chiều còn 100 thuộc tính

Sử dụng phương pháp trích xuất thuộc tính/đặc trưng *n-gram* cho bộ vector 8420 thuộc tính. Nếu đem trực tiếp bộ vector này làm tập dữ liệu đầu vào cho các thuật toán phân lớp, ta có kết quả như sau:

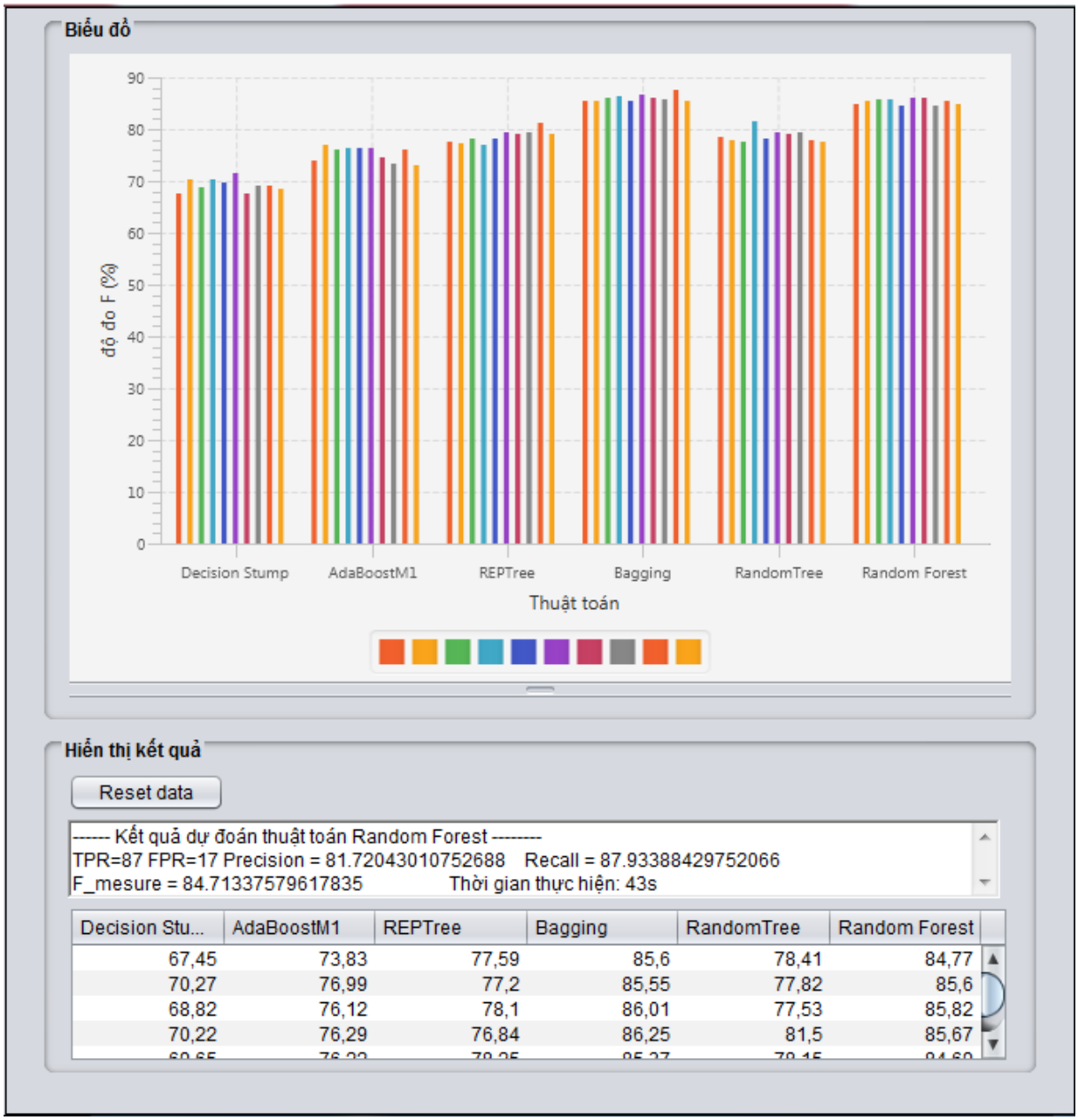
Bảng 4-6: Kết quả thực nghiệm phương pháp trích xuất thuộc tính n-gram, không giảm chiều thuộc tính

Độ đo F (%)	Decision Stump	AdaBoost	REPTree	Bagging	Random Tree	Random Forest
S1	67,45	73,83	77,59	85,60	78,41	84,77
S2	70,27	76,99	77,20	85,55	77,82	85,60
S3	68,82	76,12	78,10	86,01	77,53	85,82
S4	70,22	76,29	76,84	86,25	81,50	85,67
S5	69,65	76,22	78,25	85,37	78,15	84,69
S6	71,40	76,30	79,35	86,55	79,53	86,19
S7	67,55	74,61	78,95	86,06	79,11	86,09
S8	69,02	73,27	79,27	85,74	79,45	84,68
S9	68,98	76,01	81,10	87,54	78	85,35
S10	68,41	73,12	79,21	85,56	77,46	84,71
Trung bình	**69,18**	**75,28**	**78,59**	**86,02**	**78,70**	**85,36**

Trong đó, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10 là các bộ dữ liệu kiểm định sử dụng cho thực nghiệm. Kết quả được biểu diễn bằng giá trị độ đo F (%).

Bảng 4-7: Thời gian thực hiện phương pháp trích xuất thuộc tính n-gram, không giảm chiều thuộc tính

Thời gian (s)	Decision Stump	AdaBoost	REPTree	Bagging	Random Tree	Random Forest
n-gram – không giảm chiều thuộc tính	111;114; 105;109; 107;109; 110;110; 101;108	973;981; 962;971; 968;980; 982;979; 975;982	1165;1162; 1167;1167; 1161;1165; 1171;1168; 1175;1180	7572;7560; 7580;7582; 7570;7578; 7582;7573; 7585;7594	20;26;26; 18;18;18; 18;19;20; 25	46;77;44; 44;43;43; 45;43;46; 43
	108,4	**975,3**	**1168,1**	**7577,6**	**20,8**	**47,4**



Hình 4-8: Biểu đồ kết quả thực nghiệm phương pháp trích xuất thuộc tính n-gram, không giảm chiều số thuộc tính

Nếu đem bộ vector thuộc tính ban đầu sau khi trích xuất thuộc tính bằng phương pháp n-gram giảm chiều còn 100 thuộc tính với phương pháp MRMD, sau đó đem tập vector thuộc tính đã rút gọn làm đầu vào cho các thuật toán phân lớp. Ta thu được kết quả thực nghiệm như sau:

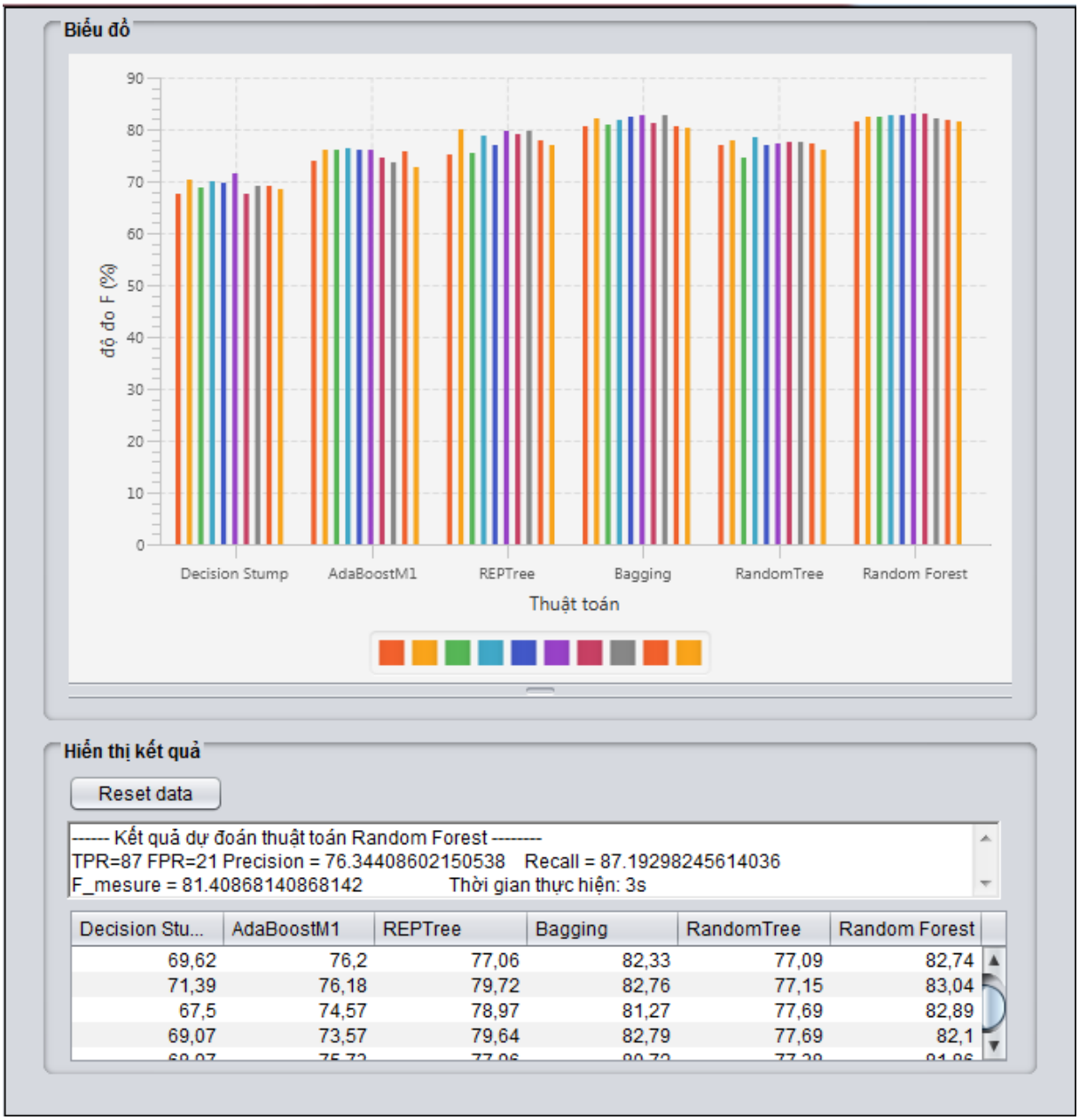
Bảng 4-8: Kết quả thực nghiệm phương pháp trích xuất thuộc tính n-gram, giảm chiều còn 100 thuộc tính

Độ đo F (%)	Decision Stump	AdaBoost	REPTree	Bagging	Random Tree	Random Forest
S1	67,45	73,89	75,14	80,52	77,07	81,54
S2	70,27	76,01	80,03	81,99	78,03	82,54
S3	68,92	76,15	75,52	80,91	74,56	82,37
S4	70,12	76,26	78,86	81,74	78,54	82,58
S5	69,62	76,20	77,06	82,33	77,09	82,74
S6	71,39	76,18	79,72	82,76	77,15	83,04
S7	67,50	74,57	78,97	81,27	77,69	82,89
S8	69,07	73,57	79,64	82,79	77,69	82,10
S9	68,97	75,73	77,96	80,72	77,38	81,86
S10	68,39	72,74	77,05	80,35	76,08	81,41
Trung bình	**69,17**	**75,13**	**78,00**	**81,54**	**77,13**	**82,31**

Trong đó, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10 là các bộ dữ liệu kiểm định sử dụng cho thực nghiệm. Kết quả được biểu diễn bằng giá trị độ đo F (%).

Bảng 4-9: Thời gian thực hiện phương pháp trích xuất thuộc tính n-gram, giảm chiều còn 100 thuộc tính

Thời gian (s)	Decision Stump	AdaBoost	REPTree	Bagging	Random Tree	Random Forest
n-gram – 100 thuộc tính	1;1;1;0,5; 0,5;0,5; 0,5;0,5; 0,5;0,5	6;6;6;6;6; 5;5;6;6;6	2;1;1;2;1; 2;2;1;2;2	14;14;15; 17;14;15; 15;15;16; 16	0,5;0,5; 0,5;0,5; 0,5;0,5; 0,5;0,5	3;3;3;3; 3;3;3;3; 3;3
	0,65	**5,8**	**1,6**	**15,1**	**0,5**	**3**



Hình 4-9: Biểu đồ kết quả thực nghiệm phương pháp trích xuất thuộc tính n-gram, giảm chiều còn 100 thuộc tính

Tổng kết, ta có bảng rút gọn kết quả thực nghiệm trong nghiên cứu cho các phương pháp trích xuất thuộc tính/đặc trưng, các phương pháp phân lớp như sau:

Bảng 4-10: Bảng kết quả tổng hợp các phương pháp phân lớp

Phương pháp	Đơn vị	Decision Stump	Ada Boost	REP Tree	Bagging	Random Tree	Random Forest
MLD – không giảm chiều thuộc tính (1134 thuộc tính)	Độ đo F (%)	67,76	70,71	79,63	86,29	77,81	85,39
	Thời gian (s)	18,6	106,8	34,9	225,6	9,1	15,6
MLD – giảm chiều còn 100 thuộc tính	Độ đo F (%)	68,17	69,46	74,92	82,44	74,60	81,58
	Thời gian (s)	2,2	7,5	2,1	17,7	1	4
n-gram – không giảm chiều thuộc tính (8420 thuộc tính)	Độ đo F (%)	69,18	75,28	78,59	86,02	78,70	85,36
	Thời gian (s)	108,4	975,3	1168,1	7577,6	20,8	47,4
n-gram – giảm chiều còn 100 thuộc tính	Độ đo F (%)	69,18	75,13	78,00	81,54	77,13	82,31
	Thời gian (s)	0,65	5,8	1,6	15,1	0,5	3

4.3 NHẬN XÉT

Về tổng quan ta nhận thấy các mô hình phân lớp đơn lẻ có độ chính xác trong kiểm định thấp hơn nhiều so với các mô hình phân lớp tổng hợp tương ứng mà sử dụng mô hình phân lớp đơn lẻ đó làm cơ sở. Cụ thể, hiệu quả dự đoán của mô hình thuật toán *Decision Stump* thấp hơn mô hình thuật toán *AdaBoostM1* trung bình khoảng 4% (theo độ đo F), hiệu quả dự đoán mô hình thuật toán *REPTree* thấp hơn mô hình thuật toán *Bagging* trung bình khoảng 7% (theo độ đo F), và hiệu quả dự đoán mô hình thuật toán *Random Tree* thấp hơn mô hình thuật toán *Random Forest* trung bình khoảng 7% (theo độ đo F). Nhưng xét về chi phí cho bài toán, các phương pháp phân lớp đơn lẻ có chi phí thấp hơn khá nhiều so với các phương pháp phân lớp tổng hợp tương ứng. Cụ thể, chi phí cho thuật toán phân lớp *Decision Stump* xấp xỉ trong khoảng [20;30] (%) chi phí cho thuật toán phân lớp *AdaBoostM1*, chi phí cho thuật toán phân lớp *REPTree* xấp xỉ trong khoảng [12;15] (%) chi phí cho thuật toán phân lớp *Bagging*, chi phí cho thuật toán phân lớp *Random Tree* xấp xỉ trong khoảng [25;50] (%) chi phí cho thuật toán phân lớp *Random Forest*.

Tiếp theo, nhận xét về hiệu quả dự đoán phân lớp khi sử dụng phương pháp lựa chọn thuộc tính/đặc trưng *MRMD* để giảm chiều dữ liệu. Ta thấy các thuật toán phân lớp sử dụng đầu vào là tập vector thuộc tính rút gọn có chi phí giảm đáng kể so với sử dụng đầu vào giữ nguyên là tập vector thuộc tính ban đầu, mức chi phí giảm trong khoảng từ [10;25](%). Nhưng hiệu quả dự đoán giảm xuống, dao động trong khoảng [1;4](%) (theo độ đo F). Mức hiệu quả dự đoán bị giảm trên có thể chấp nhận được so với chi phí chạy chương trình tiết kiệm được.

So sánh giữa hai phương pháp trích xuất thuộc tính/đặc trưng là *n-gram* và *MLD*. Ta thấy hiệu quả dự đoán và chi phí bỏ ra như sau: Hiệu quả cho 2 phương pháp trích xuất thuộc tính/đặc trưng là tương đương nhau, chi phí bỏ ra chạy thuật toán với trường hợp giảm chiều thuộc tính thì phương pháp *n-gram* tốt hơn phương pháp *MLD*, với trường hợp không giảm chiều thuộc tính thì phương pháp *n-gram* không phù hợp vì thời gian xử lý thuật toán quá lâu, đặc biệt áp dụng với thuật toán phân lớp *Bagging*. Nhưng chi phí để thực hiện giảm chiều thuộc tính cho 2 phương pháp *n-gram* và *MLD* thì phương pháp *MLD* có chi phí thấp hơn nhiều lần so với phương pháp *n-gram* cả về thời gian và cấu hình máy tính yêu cầu. Vì vậy nếu xét tính hiệu quả ta sẽ chọn *MLD* thay vì *n-gram*.

So sánh giữa các cặp thuật toán với nhau, cụ thể cặp *Decision Stump – AdaBoostM1*, *REPTree – Bagging*, và cặp *Random Tree – Random Forest*. Ta thấy cặp *Decision Stump – AdaBoostM1* có hiệu quả dự đoán thấp hơn 2 cặp còn lại. Hai cặp *REPTree – Bagging* và *Random Tree – Random Forest* có hiệu quả dự đoán tương đương nhau, nhưng xét chi phí cho thuật toán thì cặp *Random Tree – Random Forest* có chi phí bỏ ra thấp hơn nhiều lần so với cặp *REPTree – Bagging*.

Từ những nhận xét trên, ta rút ra kết quả cuối cùng: Phương pháp hiệu quả nhất trong nghiên cứu này cho dự đoán bài toán "Dự đoán tương tác protein – protein sử dụng phương pháp khai phá dữ liệu" là phương pháp phân lớp *Random Forest*, có sử dụng phương pháp trích xuất thuộc tính/đặc trưng *MLD* và phương pháp lựa chọn thuộc tính/đặc trưng *MRMD* để giảm chiều thuộc tính.

4.4 KẾT LUẬN

Luận văn đã đạt được hai kết quả quan trọng trong quá trình xây dựng chương trình dự đoán tương tác protein - protein sử dụng kỹ thuật khai phá dữ liệu.

Về nghiên cứu tìm hiểu:

- Nghiên cứu các khái niệm sinh học liên quan protein, cấu trúc protein
- Nghiên cứu các khái niệm khai phá dữ liệu nền tảng liên quan đến kỹ thuật phân lớp dữ liệu
- Tìm hiểu tổng quan về một số thuật toán phân lớp cơ bản
- Tìm hiểu về phương pháp phân lớp tổng hợp (ensemble) và một số phương pháp kết hợp các bộ phân lớp cơ bản
- Tìm hiểu các khái niệm về đánh giá mô hình phân lớp

Về thực nghiệm:

- Xây dựng được chương trình dự đoán tương tác protein - protein bằng phương pháp phân lớp tổng hợp
- Xây dựng được hàm đánh giá và so sánh kết quả thực nghiệm giữa phương pháp phân lớp tổng hợp và phân lớp đơn lẻ
- Tiến hành thử nghiệm trên nhiều tập dữ liệu ngẫu nhiên khác nhau để đảm bảo tính chính xác khách quan
- Xây dựng giao diện trực quan, dễ dàng sử dụng cho người dùng

Luận văn đã giới thiệu phương pháp áp dụng mô hình phân lớp tổng hợp vào nghiên cứu dự đoán tương tác protein - protein. Cũng như chứng minh được về mặt lý thuyết

và thực nghiệm rằng phương pháp áp dụng mô hình phân lớp tổng hợp này ưu việt hơn giải thuật mô hình phân lớp đơn lẻ, có độ chính xác cao hơn và độ ổn định tốt hơn.

So với các công trình nghiên cứu đã công bố, đóng góp của luận văn này có thêm sự so sánh giữa các bước xây dựng mô hình dự đoán phân lớp, để tìm ra phương pháp dự đoán hiệu quả nhất. Và chứng minh được rằng hầu hết các kết quả thu được từ mô hình phân loại tổng hợp là hiệu quả hơn dự đoán bằng mô hình phân loại đơn lẻ. Từ đó có thêm căn cứ nghiên cứu sâu hơn về mô hình phân loại tổng hợp áp dụng vào bài toán "Dự đoán tương tác protein – protein".

4.5 HƯỚNG NGHIÊN CỨU TRONG TƯƠNG LAI

Trong luận văn tôi chưa đi sâu vào tìm hiểu được cách kết hợp các thuật toán con trong thuật toán phân lớp tổng hợp. Về ngôn ngữ lập trình vấn đề tối ưu thời gian và hiệu suất xử lý nguồn dữ liệu lớn còn hạn chế, từ đó làm giảm độ chính xác của kết quả thực nghiệm. Vì vậy, trong tương lai, tôi mong muốn được tìm hiểu và áp dụng sâu hơn các cách kết hợp giải thuật đơn lẻ vào mô hình phân lớp tổng hợp và thực hiện tối ưu về mặt ngôn ngữ lập trình đảm bảo xử lý dữ liệu lớn một cách nhanh chóng cả về thời gian và hiệu suất xử lý.

TÀI LIỆU THAM KHẢO

[1] R. E. H. Geoffrey M. Cooper (2004). *The Cell: A Molecular Approach*, 832 pages.

[2] P. J. Chaput (2012).[online] Available at: http://www.futura-sciences.com/sante/actualites/medecine-alzheimer-parkinson-nouvelle-piste-300-maladies-35922/ [Accessed 12 September 2017]

[3] D. Whitford (2005). *Proteins: Structure and Function*, 542 pages.

[4] R. Bailey (2017). [online] Available at: https://www.thoughtco.com/protein-function-373550 [Accessed 12 September 2017]

[5] G. Filiano (2016). [online]. Available at: http://sb.cc.stonybrook.edu/news/general/2016-07-12-new-method-to-model-protein-interactions-may-help-accelerate-drug-development.php [Accessed 12 September 2017].

[6] G. Waksman (2005). *Proteomics and Protein-Protein Interactions: Biology, Chemistry, Bioinformatics, and Drug Design*, pp. 90-91.

[7] T. M. Mitchell (1997). Machine Learning. *McGraw-Hill Science/Engineering/ Math*, (March 1, 1997), pp. 3-5.

[8] I. Rish (2001). *An empirical study of the naive Bayes classifier*, pp. 2-3

[9] O. M. Lior Rokach (2008). Data mining with decision trees: theory and applications. *World Scientific Publishing Co. Pte. Ltd*, pp.4-5

[10] Zhang Q. et al (2012). *Structure-based prediction of protein-protein interactions on a genome-wide scale*, pp. 2-3.

[11] Pitre S. et al (2006). *PIPE: a protein-protein interaction prediction engine based on the re-occurring short polypeptide sequences between known interacting protein pairs*, pp. 2-3.

[12] Liu B. et al (2009). *Prediction of protein-protein interactions based on*, pp. 2-3.

[13] Urquiza J. et al (2011). *Method for Prediction of Protein-Protein Interactions in Yeast Using Genomics/Proteomics Information and Feature Selection*, pp. 2-3.

[14] Szklarczyk D. et al (2011). *The STRING database in 2011: functional interaction networks of proteins, globally integrated and scored*, pp. 2-3.

[15] Cai L. et al (2003). *SVM-Prot: web-based support vector machine software for functional classification of a protein from its primary sequence*, pp.3-4

[16] Zou Q et al (2013). Identifying Multi-Functional Enzyme by Hierarchical. *Journal of Computational & Theoretical Nanoscience,* pp. 1038-1043.

[17] Ioannis X. et al (2000). DIP: the Database of Interacting Proteins. *PubMed Central*, pp. 289-291.

[18] Philipp B. et al (2014). Negatome 2.0: a database of non-interacting proteins derived by literature mining, manual annotation and protein structure analysis. *PubMed Central*, 42:D396-D400.

[19] Liu B. et al (2008). A discriminative method for protein remote homology detection and fold recognition combining Top-n-grams and latent semantic analysis. *BMC Bioinformatics,* 9:510.

[20] Zhu-Hong Y. et al (2015). Predicting Protein-Protein Interactions from Primary Protein Sequences Using a Novel Multi-Scale Local Feature Representation Scheme and the Random Forest. *PLoS One* 10.