TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA TOÁN - CƠ - TIN HỌC

BÁO CÁO HỌC MÁY

ĐỀ TÀI

PHÂN LOẠI PROTEIN DỰA TRÊN CHUỖI AMINO ACID

GIẢNG VIÊN: CAO VĂN CHUNG

Nhóm sinh viên thực hiện

Nguyễn Hoàng Việt	**22001658**
Nguyễn Thừa Tuân	**22001652**
Lương Vân Khoa	**22001602**

Hà Nội, 2024

Lời mở đầu

Trong thế giới sinh học, protein đóng vai trò rất quan trọng, với mỗi loại có chức năng, cấu trúc và hoạt động khác nhau. Để hiểu rõ về các đặc tính và chức năng của từng loại protein, việc phân loại chúng một cách chính xác là cần thiết. Phân loại protein không chỉ giúp chúng ta hiểu được sự đa dạng của các protein mà còn hỗ trợ trong các ứng dụng như phát hiện bệnh, chẩn đoán y tế, và thiết kế thuốc.

Amino axit là đơn vị cấu trúc cơ bản của protein. Một chuỗi amino axit dài có thể chứa từ hàng chục đến hàng nghìn amino axit, tạo nên sự phức tạp trong việc xác định cấu trúc và chức năng của protein. Mỗi chuỗi amino axit có một dãy các ký tự đặc trưng, biểu diễn mã di truyền mã hóa cho protein đó. Việc phân loại protein dựa vào chuỗi amino axit đòi hỏi phải sử dụng các phương pháp tiên tiến để xác định các tính chất đặc trưng như cấu trúc, chức năng và tương tác của chúng.

Trong nghiên cứu này, mục tiêu chính là áp dụng các phương pháp học máy và phân tích dữ liệu sinh học để phân loại các protein dựa trên chuỗi amino axit. Các đặc trưng từ chuỗi amino axit như chiều dài, tính acid-base, và các tính chất hóa học khác sẽ được trích xuất và sử dụng làm đầu vào cho các mô hình máy học. Quá trình này giúp tăng cường khả năng phân loại và dự đoán chính xác các loại protein, từ đó hỗ trợ cho việc nghiên cứu sâu hơn về sinh học phân tử.

Bằng cách áp dụng các phương pháp học máy, chúng ta có thể tận dụng sức mạnh của công nghệ để phân tích và tổ chức dữ liệu protein, từ đó mở rộng kiến thức của chúng ta về sinh học phân tử và thúc đẩy sự phát triển của các ứng dụng trong y học và công nghệ sinh học.

Mục lục

Danh sách hình vẽ

Chương 1 Giới thiệu đề tài

1.1 Giới thiệu về bài toán phân loại

Phân loại là một trong những bài toán cơ bản và quan trọng trong lĩnh vực học máy (Machine Learning) và khoa học dữ liệu. Mục tiêu của bài toán phân loại là gán nhãn hoặc nhóm các đối tượng dựa trên các đặc trưng của chúng. Trong sinh học phân tử, việc phân loại protein dựa trên chuỗi amino acid đóng vai trò quan trọng trong việc hiểu cấu trúc, chức năng và cơ chế hoạt động của các protein. Điều này không chỉ giúp cải thiện kiến thức về sinh học mà còn hỗ trợ trong việc nghiên cứu và phát triển các liệu pháp điều trị bệnh, thiết kế thuốc và cải tiến công nghệ sinh học.

1.2 Một số đặc trưng của protein

Đặc trưng của protein có thể được phân loại theo các nhóm chính liên quan đến chuỗi amino acid, cấu trúc và chức năng. Các đặc trưng này rất quan trọng khi áp dụng học máy để phân loại protein.
Đặc trưng liên quan đến chuỗi amino acid (Sequence Features)

- **Thành phần amino acid:** Tần suất xuất hiện của từng amino acid trong chuỗi protein (AA Composition). Mỗi protein được cấu thành từ một dãy amino acid và các loại amino acid khác nhau có tính chất sinh học khác nhau. Do đó, tỷ lệ xuất hiện của từng amino acid có thể cung cấp thông tin quan trọng về tính chất của protein.
- **Cặp amino acid liền kề:** Sự xuất hiện của các cặp amino acid liền kề trong chuỗi (Dipeptide Composition). Những cặp amino acid này có thể giúp dự đoán cấu trúc ba chiều của protein, vì chúng thể hiện cách các amino acid tương tác với nhau trong chuỗi protein.
- **Tính chất hóa lý của amino acid:** Đo lường các tính chất hóa lý của các amino acid trong chuỗi protein, như độ phân cực (Polar/Nonpolar), tính ưa nước (Hydrophilicity/Hydrophobicity), tính acid/base (Acidic, Basic, Neutral). Các tính chất này có ảnh hưởng đến cấu trúc và chức năng của protein, ví dụ như khả năng gấp nếp và tương tác với các phân tử khác.
- **Chiều dài chuỗi amino acid:** Số lượng amino acid trong chuỗi protein có thể ảnh hưởng đến cấu trúc và chức năng của protein. Một chuỗi protein dài hơn có thể tạo ra một cấu trúc phức tạp hơn, dẫn đến một chức năng sinh học khác biệt.

Amino Acid	Three-Letter Code	One-Letter Code
alanine	ala	A
arginine	arg	R
asparagine	asn	N
aspartic acid	asp	D
cysteine	cys	C
glutamic acid	glu	E
glutamine	gln	Q
glycine	gly	G
histidine	his	H
isoleucine	ile	I
leucine	leu	L
lysine	lys	K
methionine	met	M
phenylalanine	phe	F
proline	pro	P
serine	ser	S
threonine	thr	T
tryptophan	trp	W
tyrosine	tyr	Y
valine	val	V

Bảng 1.1: Bảng các amino acid và mã của chúng

Chương 2 Xử lý dữ liệu

2.1 Tiền xử lý dữ liệu

Quá trình tiền xử lý dữ liệu trong bài toán được thực hiện nhằm đảm bảo dữ liệu đầu vào đầy đủ, sạch sẽ và có cấu trúc phù hợp để sử dụng trong các bước phân tích tiếp theo.

2.1.1 Tích hợp dữ liệu

Nếu dữ liệu lấy từ 2 bảng khác nhau, tích hợp dữ liệu là quá trình gộp 2 bảng lại bằng cách thực hiện phép nối (merge) trên cột chung. Điều này đảm bảo rằng dữ liệu cuối cùng bao gồm đầy đủ các thông tin từ cả hai nguồn.

2.1.2 Loại bỏ dữ liệu trùng lặp và thiếu

Loại bỏ dữ liệu trùng lặp và thiếu là quy trình làm sạch dữ liệu nhằm loại bỏ hoặc xử lý các bản ghi trùng lặp và các giá trị thiếu:

- Dữ liệu trùng lặp: Xác định và loại bỏ những bản ghi giống nhau để tránh sự không chính xác trong phân tích và giảm nhiễu dữ liệu.
- Dữ liệu thiếu: Xử lý các giá trị trống bằng cách loại bỏ các bản ghi không đầy đủ hoặc thay thế chúng bằng các giá trị hợp lý, giúp cải thiện chất lượng dữ liệu và hiệu quả phân tích.

2.1.3 Lọc dữ liệu theo tần suất lớp

Đây là kỹ thuật xử lý dữ liệu tập trung vào phân bố tần suất của các lớp (class distribution) trong dữ liệu. Điều này đặc biệt quan trọng đối với các bài toán phân loại (classification), nơi mà dữ liệu không cân bằng (imbalanced data) có thể gây ra sự sai lệch trong việc huấn luyện mô hình.
Các bước để lọc dữ liệu theo tần suất lớp:

- Đếm tần suất lớp: Tính số lượng mẫu n_k thuộc mỗi lớp k trong tập dữ liệu, giúp hiểu rõ mức độ phân bố dữ liệu giữa các lớp.
- Lọc dữ liệu bằng ngưỡng tần suất: Chọn ra một ngưỡng T nào đó, Loại bỏ các lớp có số mẫu bé hơn T, vì chúng có thể gây nhiễu hoặc không đủ dữ liệu để mô hình học hiệu quả.

Để chọn được giá trị T phù hợp, chúng ta có thể thực hiện theo một trong hai cách: quan sát trực tiếp hoặc sử dụng các tiêu chí thống kê.

- Quan sát trực tiếp: Có thể dựa vào quan sát trực quan số lượng mẫu của các lớp để đưa ra một đánh giá thực tế, từ đó chọn ra giá trị T.
- Sử dụng các tiêu chí thống kê: Tính toán và chọn một trong các giá trị trung bình, trung vị hay một số phần trăm nào đó trên tổng số mẫu để lấy làm giá trị T.

2.1.4 Trích xuất đặc trưng từ chuỗi dữ liệu

Đây là quá trình biến đổi dữ liệu chuỗi thành các đặc trưng số học mà các mô hình Machine Learning có thể sử dụng. Các đặc trưng này đại diện cho các thông tin quan trọng của dữ liệu chuỗi và giúp làm giảm sự phức tạp cho việc xử lý và phân tích tiếp theo.
Phương pháp CountVectorizer thường được sử dụng để biến đổi các chuỗi sequence thành các đặc trưng số học. Quá trình này áp dụng kỹ thuật n-gram.
Trong đó:

- **n-gram**: Kỹ thuật n-gram chia văn bản thành các nhóm n ký tự liên tiếp. Mỗi nhóm n ký tự được coi là một đặc trưng riêng biệt trong dữ liệu. Điều này giúp giữ lại sự tương quan giữa các ký tự gần nhau trong văn bản, như các từ trong câu hoặc các ký tự trong chuỗi.
- **CountVectorizer**: Đây là một công cụ trong `scikit-learn` dùng để chuyển đổi văn bản thành tập hợp các đặc trưng số học. CountVectorizer hoạt động theo các bước sau:
  - **Tokenization**: Nó chia văn bản thành các từ hoặc n-gram (dưới dạng các nhóm liên tiếp các ký tự). Tùy thuộc vào tham số `analyzer`, `token_pattern`, hoặc `ngram_range` mà bạn đã chỉ định.
  - **Tạo bảng tần suất**: Sau khi token hóa, nó đếm số lần mỗi từ hoặc n-gram xuất hiện trong toàn bộ văn bản. Mỗi từ hoặc n-gram được coi là một cột trong bảng tần suất.
  - **Biểu diễn dưới dạng ma trận**: Mỗi tài liệu được biểu diễn dưới dạng một hàng trong ma trận, với các cột là các từ hoặc n-gram. Giá trị trong ma trận thể hiện tần suất xuất hiện của n-gram đó trong tài liệu tương ứng.

2.1.5 Chuẩn hóa dữ liệu

Đây là quá trình thay đổi dữ liệu để đảm bảo rằng các giá trị có phạm vi và phân phối đồng nhất. Quá trình này thường đưa dữ liệu về dạng chuẩn phân phối Gauss với giá trị trung bình bằng 0 và phương sai bằng 1.
Chuẩn hóa dữ liệu giúp làm giảm ảnh hưởng của các thuộc tính có độ lệch lớn hoặc độ lớn khác nhau, từ đó giúp các thuật toán hoạt động ổn định hơn.

2.1.6 Giảm chiều dữ liệu

Giảm chiều dữ liệu là kỹ thuật quan trọng trong Machine Learning nhằm giảm số lượng thông tin dư thừa trong các feature vectors có thể có số chiều rất lớn. Việc giảm chiều giúp giảm bớt khó khăn về lưu trữ và tốc độ tính toán, từ đó cải thiện hiệu suất của mô hình. Nó cũng giúp nén dữ liệu, từ đó hỗ trợ quá trình học và dự đoán hiệu quả hơn.
Một phương pháp đơn giản nhất trong các thuật toán giảm chiều dữ liệu có tên là **Principal Component Analysis (PCA)**, tức Phân tích thành phần chính.

1. **Xác định ma trận dữ liệu:** PCA bắt đầu bằng việc tạo ra ma trận dữ liệu X, với mỗi hàng là một đối tượng và mỗi cột là một thuộc tính.
2. **Tính ma trận hiệp phương sai:** PCA tính toán ma trận hiệp phương sai của dữ liệu để xác định mối tương quan giữa các thuộc tính.
3. **Tìm các thành phần chính:** PCA xác định các "thành phần chính", là các vector mới kết hợp các thuộc tính ban đầu. Những thành phần này được sắp xếp theo mức độ quan trọng, giúp giảm chiều dữ liệu.
4. **Chiếu dữ liệu lên không gian mới:** Dữ liệu được chiếu từ không gian nhiều chiều xuống không gian ít chiều hơn, giữ lại thông tin quan trọng và loại bỏ thông tin dư thừa.

Ngoài PCA, một phương pháp khác để giảm chiều dữ liệu là **Truncated Singular Value Decomposition (Truncated SVD)**.

1. **Phân rã ma trận:** Truncated SVD bắt đầu bằng việc thực hiện phân rã ma trận X thành ba ma trận U, Σ, và V^T sao cho $X = U\Sigma V^T$, với U là ma trận chứa các vector riêng của ma trận hiệp phương sai, Σ là ma trận chéo chứa các giá trị riêng, và V^T là ma trận chứa các vector riêng của ma trận hiệp phương sai chuyển vị.

2. **Cắt giảm số chiều:** Thay vì sử dụng tất cả các giá trị riêng trong ma trận Σ, Truncated SVD chỉ giữ lại một số lượng nhỏ các giá trị lớn nhất và tương ứng với chúng là các vector riêng. Điều này giúp giảm số chiều của dữ liệu mà vẫn giữ được phần lớn thông tin quan trọng.

3. **Chiếu dữ liệu lên không gian thấp hơn:** Dữ liệu được chiếu vào không gian giảm chiều dựa trên các thành phần chính được chọn từ phân rã SVD, giúp giảm độ phức tạp tính toán trong các mô hình học máy.

2.1.7 Chia dữ liệu thành tập Train và Test

Tập dữ liệu được chia ngẫu nhiên thành hai phần :

- Tập Train: là phần dữ liệu được chọn từ tổng thể dữ liệu ban đầu, sử dụng để huấn luyện mô hình. Trong quá trình này, mô hình học cách nhận diện các mẫu, tìm hiểu các mối quan hệ giữa các đặc trưng và thực hiện điều chỉnh các tham số để tối ưu hóa kết quả đầu ra.

- Tập Test: là phần dữ liệu được giữ lại và không tham gia vào quá trình huấn luyện. Mục đích của nó là để đánh giá hiệu suất của mô hình sau khi được huấn luyện.

2.2 Phân cụm bằng K-means

2.2.1 Giới thiệu

K-means là một trong những thuật toán cơ bản nhất trong Unsupervised learning (Học không giám sát). Trong thuật toán K-means clustering, chúng ta không biết nhãn (label) của từng điểm dữ liệu. Mục đích của thuật toán này là phân dữ liệu thành các cụm (cluster) khác nhau sao cho dữ liệu trong cùng một cụm có tính chất giống nhau.
Ý tưởng đơn giản nhất về cụm (cluster) là tập hợp các điểm ở gần nhau trong một không gian nào đó (không gian này có thể có rất nhiều chiều trong trường hợp thông tin về một điểm dữ liệu là rất lớn).

2.2.2 Phân tích toán học

Mục đích cuối cùng của thuật toán phân nhóm này là: từ dữ liệu đầu vào và số lượng nhóm chúng ta muốn tìm, chỉ ra được trung tâm của mỗi nhóm và phân các điểm dữ liệu vào các nhóm tương ứng. Giả sử thêm rằng mỗi điểm dữ liệu chỉ thuộc vào đúng một nhóm.

- Các tâm cụm
$$m_1, m_2, \ldots, m_K \in R^{d\times 1}$$
- Nhãn của từng điểm dữ liệu thông qua vector one-hot $y_i = [y_{i1}, y_{i2}, \ldots, y_{iK}]$, trong đó:
  - $y_{ik} = 1$ nếu x_i thuộc cụm k, ngược lại $y_{ij} = 0$ với $j \neq k$.
  - Mỗi điểm dữ liệu chỉ thuộc một cụm, đảm bảo: $y_{ik} \in \{0, 1\}$ và $\sum_{k=1}^{K} y_{ik} = 1$.
- Sai số của một điểm dữ liệu x_i khi thuộc cụm k được viết lại dưới dạng nhãn one-hot y_{ij} (với $y_{ik} = 1$ nếu x_i thuộc cụm k):
$$y_{ik}\|x_i - m_k\|_2^2 = \sum_{j=1}^{K} y_{ij}\|x_i - m_j\|_2^2$$

- Tổng sai số trên toàn bộ dữ liệu (hàm mất mát) là:

$$L(Y, M) = \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \|x_i - m_j\|_2^2$$

Bài toán tối ưu: Tìm Y và M để tối thiểu hóa hàm mất mát $L(Y, M)$:

$$Y, M = \arg\min_{Y,M} \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \|x_i - m_j\|_2^2$$

với các ràng buộc:

- $y_{ij} \in \{0, 1\}, \forall i, j$ (một điểm chỉ thuộc một cụm).
- $\sum_{j=1}^{K} y_{ij} = 1, \forall i$ (một điểm luôn thuộc duy nhất một cụm).

Một cách đơn giản để giải bài toán này là xen kẽ giải Y và M khi biến còn lại được cố định. Đây là một thuật toán lặp, cũng là kỹ thuật phổ biến khi giải bài toán tối ưu. Chúng ta sẽ lần lượt giải quyết hai bài toán sau đây:

- **Fix** M, tìm Y:
  - Gán mỗi điểm dữ liệu x_i vào cụm có center m_j gần nhất: $j = \arg\min_j \|x_i - m_j\|_2^2$.
  - Nghĩa là mỗi điểm sẽ thuộc cụm có khoảng cách đến tâm cụm nhỏ nhất.
- **Fix** Y, tìm M:
  - Cập nhật center m_j của mỗi cụm bằng trung bình cộng các điểm trong cụm đó: $m_j = \frac{\sum_{i=1}^{N} y_{ij} x_i}{\sum_{i=1}^{N} y_{ij}}$.

Kỹ thuật tối ưu hóa xen kẽ: Lặp lại hai bước trên đến khi các center M hội tụ (không thay đổi đáng kể).

Ý nghĩa: Tên gọi "K-means" xuất phát từ việc các tâm cụm (M) chính là trung bình cộng (mean) của các điểm trong cụm tương ứng.

2.2.3 Tìm giá trị k tối ưu

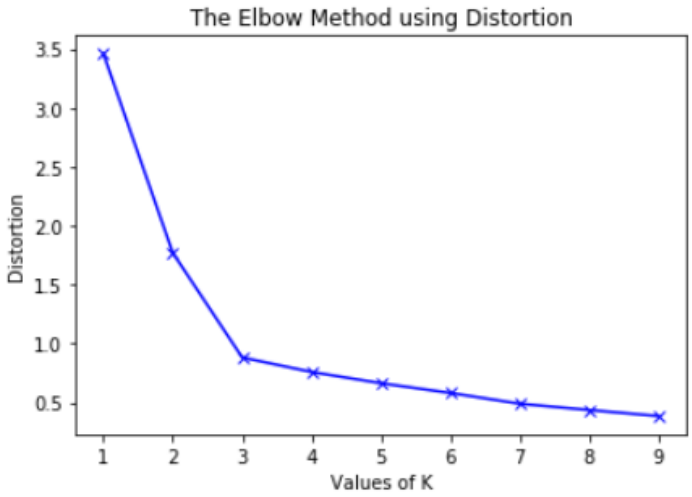
Phương pháp khuỷu tay (Elbow Method) giúp chúng ta tìm ra giá trị k tối ưu. Đây là cách hoạt động của nó:

- Chúng ta lặp lại một loạt các giá trị k, thường từ 1 đến n (n là một tham số bạn chọn).
- Với mỗi k, chúng ta tính toán Tổng bình phương trong cụm (WCSS - Within-Cluster Sum of Squares).
- WCSS đo lường mức độ tốt của các điểm dữ liệu được nhóm quanh tâm cụm tương ứng. Nó được định nghĩa là tổng bình phương khoảng cách giữa mỗi điểm và tâm cụm của nó:

$$\text{WCSS} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} \text{distance}(x_j(i), c_i)^2$$

  Trong đó, $\text{distance}(x_j(i), c_i)$ đại diện cho khoảng cách giữa điểm dữ liệu $x_j(i)$ trong cụm i và tâm cụm c_i của cụm đó.

- Điểm khuỷu tay (Elbow Point): Giá trị k tối ưu



Hình 2.1: Đồ thị phương pháp khuỷu tay (Elbow Method) cho K-Means clustering.

  - Chúng ta tính toán một thước đo khoảng cách gọi là WCSS (Tổng bình phương trong cụm). Điều này cho chúng ta biết mức độ phân tán của các điểm dữ liệu trong mỗi cụm.
  - Chúng ta thử các giá trị k khác nhau (số lượng cụm). Đối với mỗi k, chúng ta chạy KMeans và tính toán WCSS.
  - Chúng ta vẽ đồ thị với k trên trục X và WCSS trên trục Y.
  - Xác định điểm khuỷu tay: Khi chúng ta tăng k, WCSS thường giảm vì chúng ta đang tạo nhiều cụm hơn, những cụm này bắt đầu nắm bắt nhiều biến thiên của dữ liệu. Tuy nhiên, có một điểm mà việc thêm nhiều cụm chỉ làm giảm WCSS một cách không đáng kể nữa. Đây là lúc chúng ta thấy một "cú khuỷu" trong đồ thị.
  - Trước khuỷu tay: Tăng k đáng kể giảm WCSS, chỉ ra rằng các cụm mới có thể nắm bắt nhiều hơn biến thiên của dữ liệu.
  - Sau khuỷu tay: Thêm các cụm không mang lại nhiều giảm WCSS, chỉ ra rằng các cụm bổ sung có thể không cần thiết và có thể dẫn đến hiện tượng quá khớp.

Chương 3 Phương pháp phân loại

3.1 Naive Bayes

Naive Bayes là một thuật toán phân loại dựa trên Định lý Bayes, thường được dùng trong các bài toán phân loại, đặc biệt là phân loại văn bản. Thuật toán này được gọi là "Naive" vì nó đưa ra giả định rằng các đặc trưng (features) trong dữ liệu là độc lập, nghĩa là sự hiện diện hoặc vắng mặt của một đặc trưng không phụ thuộc vào các đặc trưng khác và các đặc trưng đưa vào mô hình có ảnh hưởng ngang nhau đối với đầu ra mục tiêu. Mặc dù giả định này thường không phản ánh chính xác thực tế, nhưng mô hình vẫn hiệu quả và dễ triển khai, đồng thời yêu cầu ít dữ liệu để huấn luyện so với nhiều mô hình khác.
Với hai giả thiết trên, kết hợp với việc X trong P(y|X) là vecto các đặc trưng, có thể viết dưới dạng

$$X = \{x_1, x_2, \ldots, x_n\}$$

Khi đó, kết quả mục tiêu y để P(c|X) đạt cực đại trở thành:

$$c = \arg \max_{c \in \{1,2,\ldots,C\}} P(c) \prod_{i=1}^{n} P(x_i|c)$$

Khi d lớn và các xác suất nhỏ, biểu thức trên sẽ là một số rất nhỏ, khi tính toán có thể gặp sai số. Để giải quyết việc này, nó thường được viết lại dưới dạng tương đương bằng cách lấy log của về phải:

$$c = \arg \max_{c \in \{1,\ldots,C\}} \log(p(c)) + \sum_{i=1}^{d} \log(p(x_i|c))$$

Việc này không ảnh hưởng tới kết quả vì log là một hàm đồng biến trên tập các số dương.
Naive Bayes có ba loại mô hình chính, mỗi loại phù hợp với các kiểu dữ liệu khác nhau. Tùy thuộc vào bản chất của dữ liệu, ta sẽ chọn mô hình thích hợp nhất trong số ba mô hình sau:

- Gaussian Naive Bayes
- Multinomial Naive Bayes
- Bernoulli Naive Bayes

3.1.1 Gaussian Naive Bayes

Gaussian Naive Bayes là một loại của mô hình Naive Bayes được sử dụng cho dữ liệu có giá trị liên tục và giả định rằng các đặc trưng của dữ liệu tuân theo phân phối chuẩn (Gaussian). Gaussian Naive Bayes phù hợp với các đặc trưng dạng số, liên tục, như tuổi, chiều cao, hoặc nhiệt độ, và thường được áp dụng trong các bài toán phân loại với dữ liệu có phân phối gần với phân phối chuẩn.

$$P(x_i|c) = \frac{1}{\sqrt{2\pi\sigma_{c_i}^2}} \exp\left(-\frac{(x_i - \mu_{c_i})^2}{2\sigma_{c_i}^2}\right)$$

Trong đó, bộ tham số $\theta = \{\mu_{c_i}, \sigma_{c_i}^2\}$ được xác định bởi Likelihood:

$$(\mu_{c_i}, \sigma_{c_i}^2) = \arg \max_{\mu_{c_i}, \sigma_{c_i}^2} \prod_{n=1}^{N} p\left(x_i^{(n)} \mid \mu_{c_i}, \sigma_{c_i}^2\right)$$

3.1.2 Multinomial Naive Bayes

Mô hình này chủ yếu dùng cho dữ liệu đếm dạng phân phối rời rạc, phổ biến trong các bài toán phân loại văn bản, nơi mà các đặc trưng đại diện cho số lần xuất hiện của từ trong văn bản.
Khi đó, $p(x_i \mid c)$ tỉ lệ với tần suất từ thứ i (hay feature thứ i cho trường hợp tổng quát) xuất hiện trong các văn bản của class c. Giá trị này có thể được tính bằng cách:

$$\lambda_{c_i} = p(x_i \mid c) = \frac{N_{c_i}}{N_c}$$

Trong đó:

- N_{c_i} là tổng số lần từ thứ i xuất hiện trong các văn bản của class c.
- N_c là tổng số từ (kể cả lặp) xuất hiện trong class c

Tuy nhiên, khi có từ chưa từng xuất hiện trong class c, biểu thức trên bằng 0, điều này gây sai lệch kết quả. Để khắc phục, áp dụng một kỹ thuật được gọi là *Laplace Smooth*:

$$\hat{\lambda}_{c_i} = \frac{N_{c_i} + \alpha}{N_c + d\alpha}$$

Với α là một số dương, thường bằng 1, để tránh trường hợp tử số bằng 0. Mẫu số được cộng với $d\alpha$ để đảm bảo tổng xác suất $\sum_{i=1}^{d} \hat{\lambda}_{c_i} = 1$.
Như vậy, mỗi class c sẽ được mô tả bởi bộ các số dương có tổng bằng 1: $\hat{\lambda}_c = \{\hat{\lambda}_{c_1}, \ldots, \hat{\lambda}_{c_d}\}$.

3.1.3 Bernoulli Naive Bayes

Bernoulli Naive Bayes được áp dụng khi các đặc trưng là biến nhị phân, chỉ định sự có mặt hoặc không có mặt của từ trong văn bản. Mô hình này thường được dùng cho dữ liệu dạng nhị phân
Khi đó, $p(x_i|c)$ được tính bằng:

$$p(x_i|c) = p(i|c)^{x_i} \cdot (1 - p(i|c))^{1-x_i}$$

với $p(i|c)$ có thể được hiểu là xác suất từ thứ i xuất hiện trong các văn bản của class c.

3.2 Softmax Regression

3.2.1 Giới thiệu

Với dữ liệu $\mathbf{x}$ có số chiều là $(d+1)$, trong đó có phần tử 1 được thêm vào phía trước để thể hiện hệ số tự do trong hàm tuyến tính. Hệ số tự do w_{0j} còn được gọi là **bias**.
Giả sử số lớp (classes) là C. Với kỹ thuật **one-vs-rest**, chúng ta cần xây dựng C mô hình Logistic Regression khác nhau. Các đầu ra dự đoán được tính theo hàm sigmoid:

$$a_i = \text{sigmoid}(z_i) = \text{sigmoid}(\mathbf{w}_i^T \mathbf{x})$$

Trong kỹ thuật này, các phần tử $a_i, i = 1, 2, \ldots, C$ được tính toán độc lập chỉ với z_i. Do đó, không có mối quan hệ chặt chẽ giữa các a_i, tức là tổng của chúng có thể nhỏ hơn hoặc lớn hơn 1. Nếu ta có thể khai thác được mối quan hệ giữa các z_i, kết quả của bài toán classification có thể tốt hơn.
Chú ý rằng các mô hình Linear Regression, PLA, Logistic Regression chỉ có 1 node ở output layer. Trong các trường hợp đó, tham số mô hình chỉ là một vector $\mathbf{w}$. Trong trường hợp output layer có nhiều hơn 1 node, tham số mô hình sẽ là tập hợp tham số $\mathbf{w}_i$ ứng với từng node. Lúc này, ta có ma trận trọng số:

$$\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_C]$$

3.2.2 Hàm Softmax

Công thức của Hàm Softmax

Mục tiêu là xây dựng một mô hình xác suất sao cho, với mỗi đầu vào x, a_i đại diện cho xác suất đầu vào đó thuộc về lớp i. Để thỏa mãn điều kiện này, các giá trị a_i phải dương và tổng của chúng phải bằng 1. Để đạt được điều này, ta cần xét tất cả các giá trị z_i và tính toán a_i dựa trên mối quan hệ giữa chúng.
Ngoài các điều kiện $a_i > 0$ và $\sum_i a_i = 1$, chúng ta cũng muốn một điều kiện tự nhiên: giá trị $z_i = w_i^T x$ càng lớn thì xác suất dữ liệu rơi vào lớp i càng cao. Điều này ngụ ý rằng chúng ta cần một hàm đồng biến.
Lưu ý rằng z_i có thể nhận cả giá trị âm và dương. Một hàm số mượt mà đảm bảo rằng tất cả các giá trị z_i đều dương và đồng thời đồng biến là hàm số mũ:

$$a_i = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)}, \quad \forall i = 1, 2, \ldots, C$$

Hàm này tính tất cả các giá trị a_i dựa trên tất cả các giá trị z_i, thỏa mãn tất cả các điều kiện: dương, tổng bằng 1 và giữ được thứ tự của các giá trị z_i. Hàm số này được gọi là hàm *softmax*.
Lưu ý rằng với định nghĩa này, không có giá trị a_i nào bằng 0 hoặc bằng 1, mặc dù chúng có thể rất gần 0 hoặc 1 khi z_i rất nhỏ hoặc rất lớn so với các giá trị z_j với $j \neq i$.
Lúc này, ta có thể giả sử:

$$P(y_k = i \mid x_k; W) = a_i$$

Trong đó, $P(y = i \mid x; W)$ là xác suất để một điểm dữ liệu x thuộc lớp i khi biết các tham số mô hình (ma trận trọng số) W.

3.2.3 Cross Entropy

Hàm Cross Entropy giữa hai phân phối p và q được định nghĩa là:

$$H(p, q) = E_p[-\log q]$$

Trong đó, p và q là các phân phối rời rạc (như trong bài toán của chúng ta với y và a). Đối với phân phối rời rạc, công thức này có thể viết dưới dạng:

$$H(p, q) = -\sum_{i=1}^{C} p_i \log q_i \tag{1}$$

So với hàm bình phương khoảng cách, hàm Cross Entropy có những đặc điểm quan trọng:

1. Giá trị nhỏ nhất của cả hai hàm đạt được khi $q = p$.
2. Hàm Cross Entropy phạt các giá trị q xa p rất mạnh mẽ, trong khi hàm bình phương khoảng cách không có sự phân biệt rõ ràng giữa các giá trị gần và xa nghiệm.

Cross Entropy không có tính đối xứng: $H(p, q) \neq H(q, p)$. Do đó, trong bài toán học có giám sát, p thường là đầu ra thực sự (one-hot vector), còn q là đầu ra dự đoán.

3.2.4 Hàm Mất Mát cho Softmax Regression

Hàm mất mát cho Softmax Regression đối với một điểm dữ liệu x_i và nhãn y_i được tính bằng:

$$J(W; x_i, y_i) = -\sum_{j=1}^{C} y_{ji} \log(a_{ji})$$

Trong đó, a_{ji} là xác suất dự đoán cho lớp j của điểm dữ liệu x_i.

3.2.5 Tối Ưu Hàm Mất Mát

Để tối ưu hóa hàm mất mát, chúng ta sử dụng phương pháp Stochastic Gradient Descent (SGD). Gradient của hàm mất mát đối với ma trận trọng số W cho một điểm dữ liệu x_i được tính bằng:

$$\frac{\partial J_i(W)}{\partial W} = x_i e_i^T$$

Trong đó $e_i = a_i - y_i$ là sai số dự đoán.

Chương 4 Cài đặt đánh giá thực nghiệm

4.1 Cài đặt thử nghiệm

4.1.1 Mô tả bộ dữ liệu

Dữ liệu gồm 2 file csv pdb_data_no_dups và pdb_data_seq có cột chung là "structureId". Dùng lệnh pd.merge() để ghép thông qua cột chung này và kiểu hợp nhất là "inner" để giữ lại chỉ những hàng có cột chung là "structureId", ta có được bộ dữ liệu hoàn chỉnh.
Các đặc trưng từ thông tin về protein ta lấy được từ tập dữ liệu

- structureId: ID cấu trúc
- chainId: ID chuỗi
- sequence: Trình tự axit amin
- residueCount_x: Số lượng dư lượng (góc nhìn thứ nhất)
- macromoleculeType_x: Loại đại phân tử (góc nhìn thứ nhất)
- classification: Phân loại cấu trúc
- experimentalTechnique: Kỹ thuật thí nghiệm
- macromoleculeType_y: Loại đại phân tử (góc nhìn thứ hai)
- residueCount_y: Số lượng dư lượng (góc nhìn thứ hai)
- resolution: Độ phân giải (Ångström)
- structureMolecularWeight: Khối lượng phân tử của cấu trúc
- crystallizationMethod: Phương pháp kết tinh
- crystallizationTempK: Nhiệt độ kết tinh (Kelvin)
- densityMatthews: Hệ số Matthews (tỉ lệ dung môi)
- densityPercentSol: Phần trăm dung môi trong tinh thể
- pdbxDetails: Chi tiết bổ sung
- phValue: Giá trị pH khi kết tinh hoặc chuẩn bị cấu trúc
- publicationYear: Năm công bố

4.1.2 Lọc dữ liệu theo tần suất lớp

Thông qua quan sát trực quan số lượng mẫu, chọn giá trị T = 5000 làm ngưỡng giới hạn. Tức là chỉ giữ lại những lớp có số lượng mẫu lớn hơn 5000, những lớp có số mẫu bé hơn 5000 sẽ bị loại.

4.1.3 Trích xuất đặc trưng

CountVectorizer từ sklearn.feature_extraction.text được sử dụng để chuyển đổi chuỗi văn bản thành ma trận số mà mỗi hàng đại diện cho một chuỗi, và mỗi cột đại diện cho một 4-gram ký tự. Điều này giúp capture đặc điểm ngữ nghĩa của chuỗi sequence bằng cách chỉ ra các pattern ký tự phổ biến. Sử dụng ngram_range=(4, 4) đảm bảo rằng chỉ có các 4-gram ký tự được tạo ra từ mỗi chuỗi sequence, cho phép mô hình hiểu rõ hơn về cấu trúc và đặc trưng của chuỗi.

4.1.4 Chia tập Train-Test

Dữ liệu được chia thành hai phần: 80% cho tập huấn luyện (training set) và 20% cho tập kiểm tra (test set), sử dụng hàm train_test_split từ thư viện sklearn với tham số test_size=0.2 và random_state=1 để đảm bảo tái lập kết quả trong các lần chạy khác nhau.

4.1.5 Chuẩn hóa dữ liệu

Sử dụng StandardScaler chuẩn hóa dữ liệu bằng cách làm mean của mỗi đặc trưng bằng 0 và độ lệch chuẩn bằng 1. Khi with_mean=False, nó giữ lại phạm vi giá trị ban đầu của dữ liệu nhưng điều chỉnh độ lệch chuẩn. Điều này giúp dữ liệu trở nên ổn định hơn và cải thiện hiệu suất của các mô hình học máy.

4.1.6 Giảm chiều dữ liệu

Sử dụng PCA từ thư viện sklearn.decomposition để giảm chiều dữ liệu từ X_scaled xuống còn 3 chiều. Việc này giúp tạo ra không gian mới với 3 thành phần chính, giữ lại phần lớn thông tin quan trọng của dữ liệu ban đầu, từ đó làm giảm độ phức tạp của dữ liệu và cải thiện hiệu suất mô hình sau khi huấn luyện.

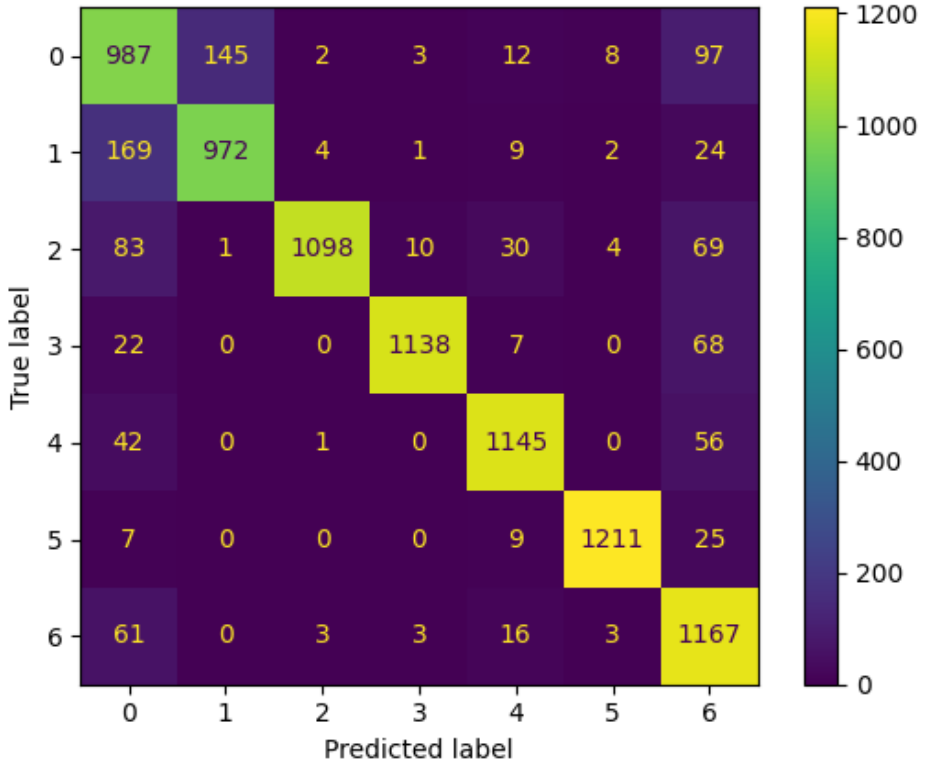
4.2 Đánh giá thực nghiệm

4.2.1 Multinomial Naive Bayes

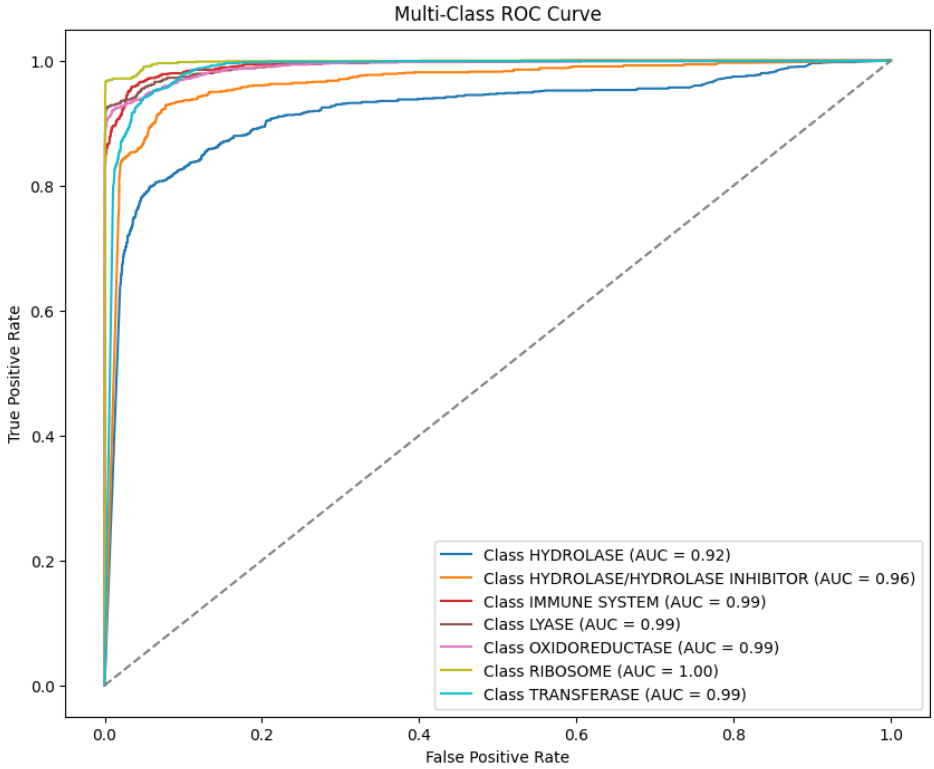
Class	**Precision**	**Recall**	**F1-score**	**Support**
HYDROLASE	0.72	0.79	0.75	1254
HYDROLASE/HYDROLASE INHIBITOR	0.87	0.82	0.85	1181
IMMUNE SYSTEM	0.99	0.85	0.91	1295
LYASE	0.99	0.92	0.95	1235
OXIDOREDUCTASE	0.93	0.92	0.93	1244
RIBOSOME	0.99	0.97	0.98	1252
TRANSFERASE	0.77	0.93	0.85	1253
Accuracy			0.89	8714
Macro avg	0.89	0.89	0.89	8714
Weighted avg	0.89	0.89	0.89	8714

Bảng 4.1: Kết quả đánh giá mô hình Multinomial Naive Bayes trên các lớp

- **Độ chính xác trên tập huấn luyện:** 93,47%
- **Độ chính xác trên tập kiểm tra:** 88,57%



Hình 4.1: Confusion matrix



Hình 4.2: ROC-Curve

- **Cross Validation Score:** 97,63%

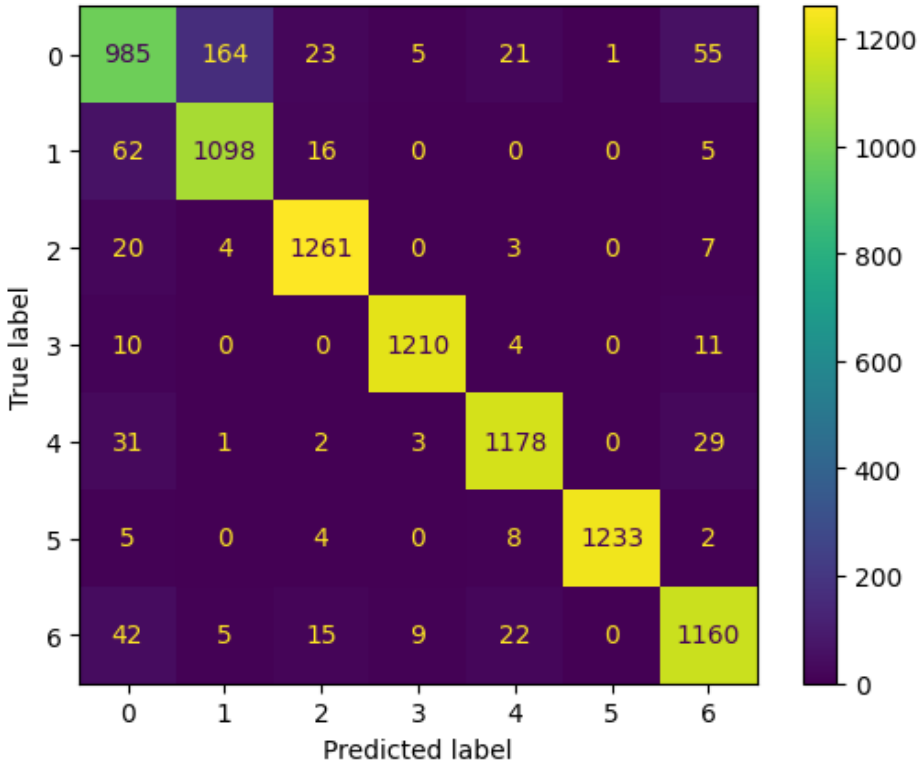
- **ROC AUC Score:** 97,75%

Kết quả trên cho thấy mô hình Multinomial Naive Bayes đạt được hiệu quả cao với độ chính xác (accuracy) đạt 89%, cùng với các chỉ số đánh giá khác như F1-score và recall đều đạt mức cao trên các lớp. Mô hình cũng thể hiện độ ổn định cao khi đạt điểm AUC ROC và điểm cross-validation tốt.

4.2.2 Softmax Regression

Class	Precision	Recall	F1-score	Support
HYDROLASE	0.85	0.79	0.82	1254
HYDROLASE/HYDROLASE INHIBITOR	0.86	0.93	0.90	1181
IMMUNE SYSTEM	0.95	0.97	0.96	1295
LYASE	0.99	0.98	0.98	1235
OXIDOREDUCTASE	0.95	0.95	0.95	1244
RIBOSOME	1.00	0.98	0.99	1252
TRANSFERASE	0.91	0.93	0.92	1253
Accuracy			0.93	8714
Macro avg	0.93	0.93	0.93	8714
Weighted avg	0.93	0.93	0.93	8714

Bảng 4.2: Kết quả đánh giá mô hình Softmax trên các lớp



Hình 4.3: Confusion matrix

- **Độ chính xác trên tập huấn luyện:** 98.35%
- **Độ chính xác trên tập kiểm tra:** 93.24%

Mô hình Softmax Regression đạt độ chính xác cao trên cả tập huấn luyện (98.35%) và tập kiểm tra (93.24%), cho thấy khả năng tổng quát tốt của mô hình. Các chỉ số Precision, Recall và F1-score đều

cao, đặc biệt là đối với lớp "RIBOSOME" với precision đạt 1.00 và recall là 0.98, cho thấy mô hình phân loại rất tốt đối với lớp này. Tuy nhiên, lớp "HYDROLASE" có recall thấp hơn, cần cải thiện thêm.

Chương 5 Kết luận và hướng phát triển

5.1 Kết luận

Kết quả thực nghiệm cho thấy cả hai mô hình Multinomial Naive Bayes và Softmax Regression đều chứng tỏ được tính khả thi trong việc giải quyết bài toán. Mô hình Softmax Regression có độ chính xác và khả năng tổng quát tốt hơn, trong khi Multinomial Naive Bayes thể hiện tính ổn định với các chỉ số đánh giá đáng tin cậy. Tuy nhiên, vẫn cần cải thiện khả năng nhận diện đối với một số lớp cụ thể như "HYDROLASE".

5.2 Hướng phát triển

Trong tương lai, để nâng cao hiệu quả và mở rộng phạm vi ứng dụng của nghiên cứu, một số định hướng phát triển quan trọng có thể được triển khai như sau:

5.2.1 Nâng cao độ chính xác của mô hình

Để nâng cao độ chính xác của mô hình, có thể thực hiện theo những hướng như sau:

- Cải thiện độ chính xác của 3 lớp HYDROLASE,HYDROLASE/HYDROLASE INHIBITOR và TRANSFERASE
- Tối ưu hóa thuật toán: Tiếp tục cải tiến các thuật toán học máy hiện tại bằng cách điều chỉnh các siêu tham số và thử nghiệm với các phương pháp tối ưu hóa mới nhằm cải thiện độ chính xác của mô hình.
- So sánh với các thuật toán học máy khác: Áp dụng các thuật toán như support vector machine.
- Sử dụng mô hình kết hợp (Ensemble Learning): Áp dụng các phương pháp như Random forest, Boosting,.. để kết hợp nhiều mô hình và khai thác sức mạnh của từng mô hình trong việc dự đoán.
- Thử nghiệm các mô hình nâng cao: Triển khai các mô hình học sâu như Mạng nơ-ron tích chập (CNN).
- Xử lý dữ liệu dạng ma trận thưa khi giải mã chuỗi protein.

5.2.2 Mở rộng dữ đoán các loại protein khác

- Dự đoán những protein ít phổ biến hơn
- Xử lý vấn đề mất cân bằng dữ liệu

TÀI LIỆU THAM KHẢO

Ngoài những slide bài giảng trên lớp, nhóm có tham khảo thêm các tài liệu:

- **Dữ liệu trên Kaggle:** https://www.kaggle.com/datasets/shahir/protein-data-set/data
- **Phân rã giá trị suy biến (SVD):** https://machinelearningcoban.com/2017/06/07/svd/
- **Phương pháp Elbow và Silhouette trong K-means:** https://bigdatauni.com/tin-tuc/cac-phuong-phap-danh-gia-trong-thuat-toan-clustering.html