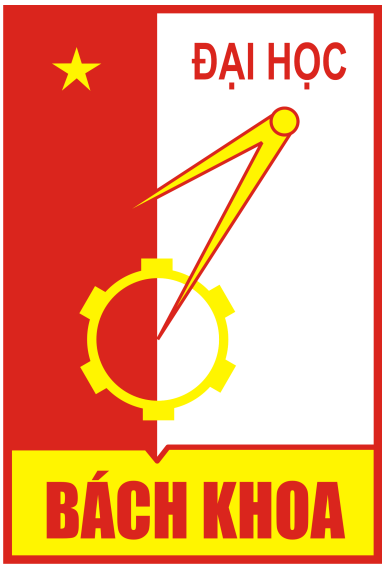
Đại học Bách khoa Hà Nội
Trường công nghệ Thông tin và Truyền thông



Bài tập lớn

Môn: Nhập môn Học máy và Khai phá dữ liệu

Tên đề tài: Dự đoán loài chim cánh cụt ở quần đảo Palmer

Giáo viên hướng dẫn: PGS. TS. Thân Quang Khoát

Nhóm sinh viên thực hiện:

Tạ Hữu Bình	20190094
Trần Trọng Hiệp	20190051
Nguyễn Văn Trung	20190071

Mã lớp: 131404

Mã HP: IT3190

Hà Nội, 10 tháng 7 năm 2022

Mục lục

Lời mở đầu

Học máy hay *machine learning* (một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các kĩ thuật với mục đích giúp cho các hệ thống "học" tự động từ dữ liệu để giải quyết những vấn đề cụ thể) đang có sự phát triển vượt bậc trong những năm gần. Không chỉ ở các nước có nền kinh tế phát triển trong top đầu của thế giới mà ngay tại các quốc gia đang phát triển, trong đó có Việt Nam, ứng dụng của học máy cũng đang xuất hiện ngày càng nhiều, đem đến những tác động tích cực to lớn cho xã hội. Vậy nên, trong bài tập lớn của học phần "Nhập môn Học máy và khai phá dữ liệu", nhóm đã lựa chọn đề tài **"Dự đoán loài chim cánh cụt ở quần đảo Palmer"**. Một số mô hình học máy, bao gồm cả mô hình *có giám sát* và mô hình *không giám sát*, sẽ được áp dụng để giải bài toán này và kiểm tra độ hiệu quả thông qua các thực nghiệm.

Báo cáo bao gồm 5 phần chính như sau:

- **Phần 1:** Giới thiệu về Machine Learning và Data Mining nói chung và bài toán "Dự đoán loài chim cánh cụt ở quần đảo Palmer"
- **Phần 2:** Trình bày về dữ liệu sử dụng và quá trình tiền xử lý dữ liệu
- **Phần 3:** Trình bày các mô hình Học máy được đề xuất để giải quyết bài toàn
- **Phần 4:** Trình bày các kết quả thực nghiệm.
- **Phần 5:** Tổng kết lại các đóng góp của báo cáo và nêu hướng phát triển trong tương lai.

.

1 Giới thiệu chung

Học máy là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các kĩ thuật cho phép các hệ thống "học" tự động từ dữ liệu để giải quyết những vấn đề cụ thể [1]. Ví dụ như các máy có thể "học" cách phân loại thư điện tử xem có phải thư rác hay không và tự động xếp thư vào thư mục tương ứng. Ngày nay, học máy được áp dụng rộng rãi bao gồm máy truy tìm dữ liệu, chẩn đoán y khoa, phát hiện thẻ tín dụng giả, phân tích thị trường chứng khoán, phân loại các chuỗi DNA, nhận dạng tiếng nói và chữ viết, dịch tự động, chơi trò chơi và cử động rô-bốt. Khai phá dữ liệu (data mining) là quá trình tính toán để tìm ra các mẫu trong các bộ dữ liệu lớn liên quan đến các phương pháp tại giao điểm của máy học, thống kê và các hệ thống cơ sở dữ liệu [2]. Mục tiêu tổng thể của quá trình khai thác dữ liệu là trích xuất thông tin từ một bộ dữ liệu và chuyển nó thành một cấu trúc dễ hiểu để sử dụng tiếp.

Một trong những nhóm bài toán phổ biến nhất của học máy là phân loại đa lớp, tiếng anh là *multiclass classification*. Với mục đích tìm hiểu, thử nghiệm, và so sánh các mô hình (model) và kỹ thuật (technique) phân loại đa lớp khác nhau, nhóm đã lựa chọn bài toán "dự đoán loài chim cánh cụt" (Ảnh ba loài chim được cho trong Hình 1) dựa trên bộ dữ liệu dạng bảng của các đặc tính liên quan đến ba loài chim cánh cụt khác nhau ở quần đảo Palmer.

Hình 1: Ba loài chim cánh cụt trên quần đảo Palmer [3].

Quá trình giải quyết bài toán sẽ được thực hiện qua các giai đoạn sau:

- Chuẩn bị tập dữ liệu huấn luyện (training dataset) và rút trích đặc trưng (feature extraction): Đây được coi là một trong những công đoạn quan trọng nhất, là đầu vào cho việc học để tìm ra mô hình giải quyết bài toán. Chúng ta phải biết cần chọn ra những đặt trưng tốt (good feature) của dữ liệu; lược bỏ những đặc trưng không tốt của dữ liệu, gây nhiễu (noise); hay xử lý các thông tin bị thiếu. Bước này cũng chuẩn bị dữ liệu để huấn luyện và đo đạc chất lượng mô hình, bao gồm tập huấn luyện (training)

để huấn luyện mô hình, tập xác thực (validation) để kiểm định độ hiệu quả trong quá trình huấn luyện và tập kiểm tra (test) để đo chất lượng của mô hình trong thực tế.

- Xây dựng mô hình phân lớp (classifier model): Mục đích của bước này là xây dựng các mô hình xác định một tập các lớp dữ liệu. Thông thường để xây dựng mô hình phân lớp cần sử dụng các thuật toán học có giám giám sát như K-nearest Neighbor hay Naive Bayes,... Báo cáo này sẽ trình bày thêm cả cách tiếp cận theo hướng học không giám sát thông qua thuật toán k-means Clustering, nhắm chia dữ liệu huấn luyện ra thành 3 cụm khác nhau. Mục đích là kiểm tra xem nếu chỉ biết trước dữ liệu có 3 nhóm khác nhau, mô hình này có thể nhận biết và chia thành 3 nhóm gần với thực tế hay không.
- Tiến hành các thực nghiệm để đánh giá các mô hình. Tham số chính là độ chính xác của các dữ đoán. Quá trình lựa chọn tham số thích hợp của từng mô hình sẽ được thực hiện qua tập huấn luyện và tập xác thực. Sau đó mô hình đại diện của các mô hình đề xuất sẽ được so sánh với nhau trên tập kiểm tra. Ngoài ra, một số cách tiền xử lý dữ liệu cũng được so sánh để kiểm tra độ hiệu quả.

  Các giai đoạn này sẽ được trình bày rõ hơn ở những phần sau trong báo cáo.

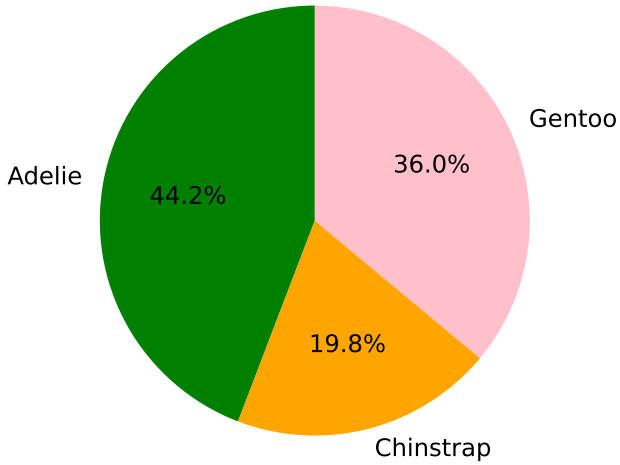
2 Dữ liệu và tiền xử lý dữ liệu

2.1 Dữ liệu

Báo cáo sử dụng dữ liệu về chim cánh cụt ở quần đảo Palmer (thuộc lục địa Nam Cực) từ trang web Kaggle [4]. Dữ liệu gốc lần đầu được công bố bởi Kristen B. Gorman và cộng sự trong một bài báo đăng trên tạp chí PlOS One [5].

Dữ liệu chứa thông tin về ba loài chim cánh cụt khác nhau. Số lượng loài Adelie đang đông nhất với 44.2 %, tiếp ngay sau là Gentoo với 36.0 % (Hình 2). Các thuộc tính xét đến gồm có:

1. Tên dự án nghiên cứu (tên cột tương ứng trong dữ liệu: **studyName**)
2. Số định danh của mẫu ứng với từng dự án (tên cột tương ứng trong dữ liệu: **Sample Number**)
3. Loài chim (tên cột tương ứng trong dữ liệu: **Species**)
4. Vùng sinh sống (tên cột tương ứng trong dữ liệu: **Region**)
5. Đảo (tên cột tương ứng trong dữ liệu: **Island**)



Hình 2: Phân bố theo số lượng của từng loài chim cánh cụt.

6. Đặc điểm về thời kỳ trưởng thành (tên cột tương ứng trong dữ liệu: **Stage**)
7. Định danh (tên cột tương ứng trong dữ liệu: **Individual ID**)
8. Khả năng nở của trứng (tên cột tương ứng trong dữ liệu: **Clutch Completion**)
9. Ngày trứng được sinh (tên cột tương ứng trong dữ liệu: **Date Egg**)
10. Chiều dài sống mỏ (tên cột tương ứng trong dữ liệu: **Culmen Length (mm)**)
11. Độ sâu sống mỏ (tên cột tương ứng trong dữ liệu: **Culmen Depth (mm)**)
12. Độ dài cánh (tên cột tương ứng trong dữ liệu: **Flipper Length (mm)**)
13. Khối lượng (tên cột tương ứng trong dữ liệu: **Body Mass (g)**)
14. Giới tính (tên cột tương ứng trong dữ liệu: **Sex**)
15. Chỉ số Nitơ trong máu (tên cột tương ứng trong dữ liệu: **Delta 15 N (o/oo)**)
16. Chỉ số Carbon trong máu (tên cột tương ứng trong dữ liệu: **Delta 13 C (o/oo)**)
17. Nhận xét (tên cột tương ứng trong dữ liệu: **Comments**)

2.2 Tiền xử lý dữ liệu

Với mục tiêu phân loại chim cánh cụt dựa trên các đặc tính sẵn có, một số cột sẽ không được xét đến, bao gồm:

- **studyName**, lý do là mỗi dự án chỉ nghiên cứu đúng một loài chim cánh cụt. Đây là đặc tính do con người tạo ra và dễ dàng suy ra được loài chim cánh cụt tương ứng.

- **Sample Number**, lý do là thuộc tính này đi kèm với từng dự án.
- **Region**, lý do là tất cả các giá trị đều giống nhau, không chứa thông tin ý nghĩa.
- **Island**, lý do là thông tin này chứa tên những đảo chỉ ứng với một loài chim cánh cụt, đây là thuộc tính ngoại cảnh.
- **Stage**, lý do là tất cả các giá trị đều giống nhau, không chứa thông tin ý nghĩa.
- **Individual ID**, đây là thông tin chủ quan do con người đặt.
- **Date Egg**, lý do là thời gian của các nghiên cứu cho từng loài chim cánh cụt có sự cách nhau, thuộc tính này có thể tiết lộ luôn loài chim cánh cụt tương ứng trong dữ liệu đang xét.
- **Comments**, lý do vì nhận xét của con người mang tính chủ quan cao, không phù hợp với mục tiêu của bài toán đặt ra.

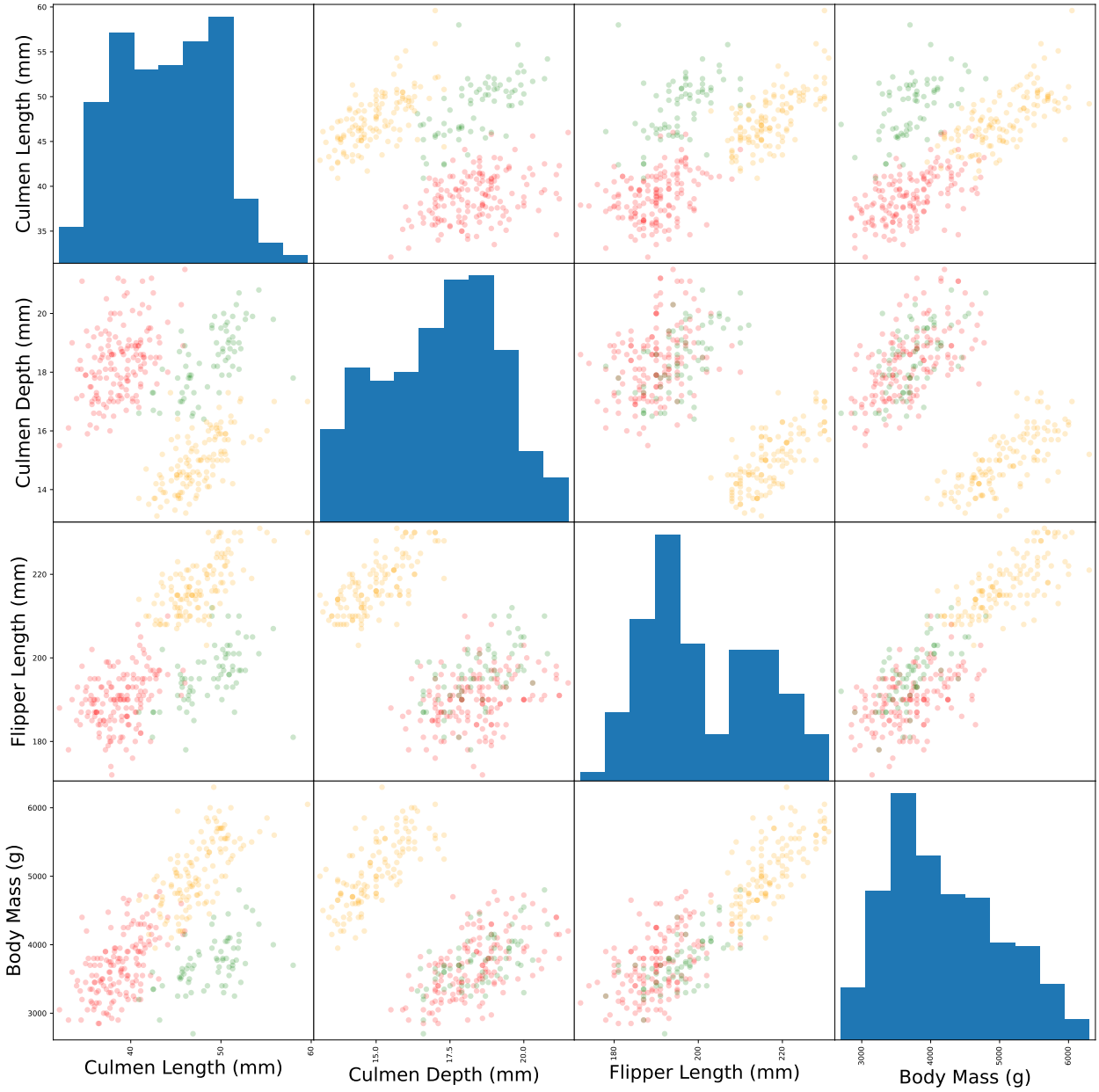
Các giá trị trong cột **Species** sẽ được dùng làm các nhãn. Những cột chứa thuộc tính còn lại được liệt kê trong Bảng 1.

Cột	Số giá trị phân biệt	Số mẫu bị thiếu
Culmen Length (mm)	164	0
Culmen Depth (mm)	80	0
Flipper Length (mm)	55	0
Body Mass (g)	94	0
Sex	2	9
Delta 15 N (o/oo)	330	12
Delta 13 C (o/oo)	331	11

Bảng 1: Một số thông tin về các đặc tính được xét đến.

Tuy nhiên, các mẫu không chứa thông tin của 6 trên 7 đặc tính trên (ngoại trừ "giới tính") sẽ bị loại bỏ khỏi bộ dữ liệu vì chứa quá ít thông tin. Tổng cộng sẽ còn lại 342 mẫu, tất cả đều khác nhau đôi một. Từ bộ dữ liệu đã xử lý, 30 % số mẫu sẽ được dùng làm bộ dữ liệu kiểm tra. Số mẫu của bộ dữ liệu xác thực chiếm 20 %, còn bộ dữ liệu huấn luyện được chia thành 3 loại với số lượng mẫu khác nhau (50, 110, và 170). Hình 3 biểu diễn ma trận đồ thị phân tán của giá trị bốn thuộc tính (chiều dài và độ sâu của sống mỏ, độ dài cánh, cân nặng) của tất cả các mẫu của cả ba loài chim cánh cụt. Có thể thấy đặc tính liên quan đến chiều dài của sống mỏ đang có sự phân tách dữ liệu rất tốt.

Ở mỗi bộ dữ liệu, các thông số dạng chữ viết sẽ được "số hóa". Dữ liệu bị thiếu ở cột **Sex** sẽ được điền giá trị đặc biệt, còn ở các cột còn lại điền giá trị trung bình ứng với tập huấn luyện (training), xác thực (valid), và kiểm tra (test); chứ không phải trên tất cả các



Hình 3: Ma trận biểu đồ phân tán của bốn thuộc tính (Chiều dài và độ sâu của sống mỏ, độ dài cánh, cân nặng), màu đỏ (Adelie), màu xanh (Chinstrap), màu cam (Gentoo).

mẫu dữ liệu. Ngoài ra, bộ dữ liệu chuẩn hóa cũng được tạo ra theo cách sau: Tính giá trị kỳ vọng và độ lệch chuẩn của các giá trị mỗi đặc tính trong bộ huấn luyện, sau đó chuẩn hóa cả bộ dữ liệu huấn luyện, xác thực và kiểm tra theo giá trị kỳ vọng và độ lệch chuẩn đó theo công thức:

$$x_{\text{mới}} = \frac{x_{\text{cũ}} - \mu}{\sigma}$$

Trong đó, x đại diện cho một giá trị ứng với đặc tính trong một bộ dữ liệu bất kỳ, μ

và σ lần lượt là giá trị kỳ vọng và phương sai của các giá trị trong cột đặc tính tương ứng của bộ dữ liệu huấn luyện.

3 Mô hình Học máy đề xuất

Phần này của báo cáo sẽ trình bày nội dung lý thuyết của bốn mô hình học máy có giám sát (K-nearest Neighbor, Random Forest, Naive Bayes, Artificial Neural Network) và một mô hình học máy không giám sát (K-Means Clustering).

3.1 K-nearest Neighbors

K-nearest neighbors là một trong những thuật toán supervised-learning đơn giản nhất (mà hiệu quả trong một vài trường hợp) trong Machine Learning. Khi huấn luyện, thuật toán này không học một điều gì từ dữ liệu huấn luyện (đây cũng là lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới.

Với KNN, trong bài toán Classification, nhãn (label) của một điểm dữ liệu mới được suy ra trực tiếp từ k điểm dữ liệu gần nhất trong tập huấn luyện. Nhãn của một dữ liệu kiểm thử có thể được quyết định bằng *major voting* (bầu chọn theo số phiếu) giữa các điểm gần nhất, hoặc nó có thể được suy ra bằng cách đánh trọng số khác nhau cho mỗi trong các điểm gần nhất đó rồi suy ra nhãn.

Có hai thành phần chính trong KNN là: lựa chọn số hàng xóm và cách tính khoảng cách giữa hai điểm dữ liệu. Trong thực tế, nên sử dụng số hàng xóm hơn để dự đoán ($k > 1$), nhưng không nên chọn quá nhiều. Nếu chọn quá nhiều hàng xóm, thì sẽ bị phá vỡ cấu trúc vốn có của dữ liệu, và do đó, dự đoán có thể không tốt. Trong không gian một chiều, khoảng cách giữa hai điểm là trị tuyệt đối giữa hiệu giá trị của hai điểm đó. Trong không gian nhiều chiều, khoảng cách giữa hai điểm có thể được định nghĩa bằng nhiều hàm số khác nhau. Trong KNN, đo lường khoảng cách đóng vai trò rất quan trọng. Một số công thức thường dùng để đo lường khoảng cách như: Minkowski ($L_p - norm$), Manhattan ($L_1 - norm$), Euclid ($L_2 - norm$), ...

Trong KNN, việc chuẩn hoá các thuộc tính đôi khi rất quan trọng để có được khả năng dự đoán tốt. Nếu dự liệu không được chuẩn hoá, thì rất có thể độ lớn của một thuộc tính nào đó có thể đóng vai trò quan trọng và lấn át một cách giả tạo các thuộc tính khác.

KNN cho bài toán Classification

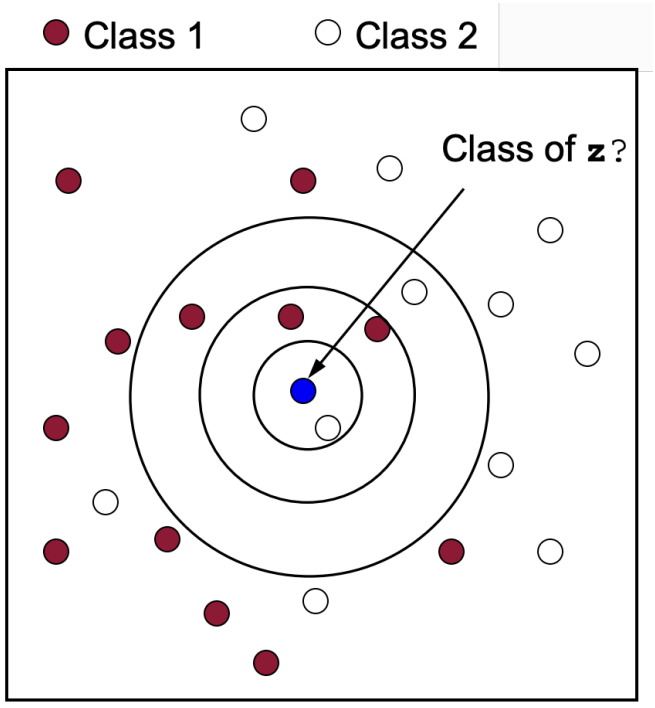
- Biểu diễn dữ liệu:

- Mỗi đối tượng quan sát được thể hiện bởi một vector n chiều, mỗi chiều đại diện cho một thuộc tính đối tượng. Ví dụ $x_i = (x_{i1}, x_{i2}, ..., x_{in})^T$
- Có tập C là tập các nhãn đã được gán trước

  - Giai đoạn học: Lưu tập dữ liệu huấn luyện D và các nhãn của nó

  - Dự đoán đầu vào mới z
- Mỗi dữ liệu x trong D, tính khoảng cách tương đồng của nó với z
- Tìm tập NB(z) là tập các hàng xóm gần với z nhất
- Sử dụng các nhãn của các dữ liệu trong NB(z) để dự đoán z

Hình 4 là mô tả ví dụ về bài toán phân lớp. Trong đó, cần phải xác định điểm dữ liệu z thuộc lớp 1 hay lớp 2.



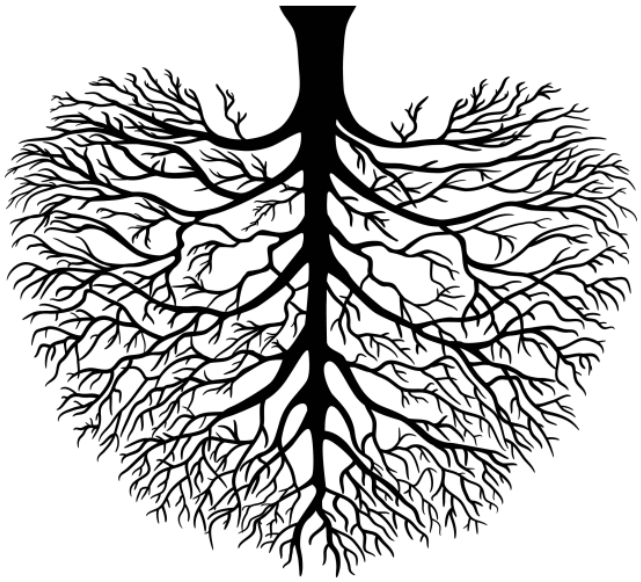
Hình 4: Ví dụ về bài toán KNN.

3.2 Random Forest

Thuật toán Random Forest (Rừng ngẫu nhiên) được tạo nên bởi kết hợp các cây quyết định (Decision Tree). Nên trước khi đề cập đến Random Forest, báo cáo sẽ trình bày về Decision Tree.

3.2.1 Decision Tree

Mô hình cây quyết định là một mô hình được sử dụng khá phổ biến và hiệu quả trong cả hai lớp bài toán Classification và Regression của học có giám sát. Khác với những thuật toán khác trong học có giám sát, mô hình cây quyết định không tồn tại phương trình dự báo. Mọi việc chúng ta cần thực hiện đó là tìm ra một cây quyết định dự báo tốt trên tập train và sử dụng cây quyết định này dự báo trên tập kiểm tra.



Hình 5: Minh họa cây quyết định.

Mỗi nhánh của cây giống như nhánh if, else nên có thể hiểu cây quyết định là tập hợp các luật nếu thì.

Bài toán Classification trong decision tree

- Biểu diễn dữ liệu:

- Mỗi đối tượng quan sát được thể hiện bởi một vector n chiều, mỗi chiều đại diện cho một thuộc tính đối tượng. Ví dụ $x_i = (x_{i1}, x_{i2}, ..., x_{in})^T, x_{i1} \in \{high, nomal\}$, $x_{i2} \in \{male, female, other\}$
- Có tập C là tập các nhãn đã được gán trước

  - Chúng ta phải học một hàm từ tập dữ liệu huấn luyện:

  $\mathbf{D} = \{(x_1, y_1), (x_2, y_2), ..., (x_M, y_M)\}$

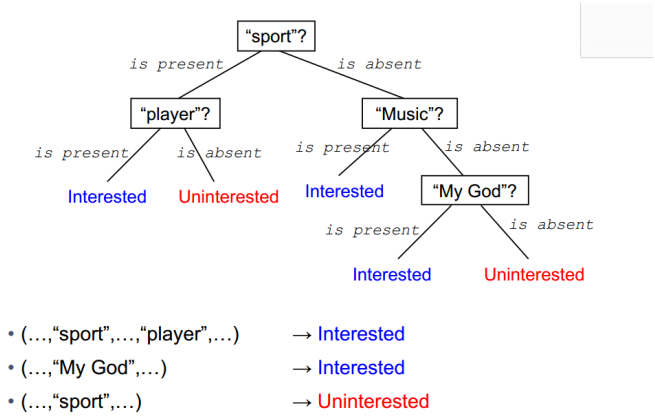
  - Mỗi đỉnh bên trong cây biểu diễn một thuộc tính để kiểm tra dữ liệu đến về sau

  - Mỗi nhánh xuất phát từ một đỉnh tương ứng với mỗi một giá trị trong miền giá trị

của thuộc tính đó

- Mỗi lá lưu nhãn lớp

- Cho dữ liệu cần phán đoán duyệt qua cây đấy, dùng các thuộc tính của nó duyệt từ gốc đến lá, lấy nhãn ở lá để phán đoán cho dữ liệu mới



Hình 6: Ví dụ về cây quyết định.

Entropy

Entropy của 1 tập S với c lớp được định nghĩa:

$$Entropy(S) = -\sum_{i=1}^{c} p_i log_2 p_i$$

Ý nghĩa của Entropy trong lý thuyết thông tin:

- Entropy cho biết số bit trung bình để mã hóa một lớp S.

- Entropy của một tin nhắn đo lượng thông tin trung bình có trong tin nhắn đó.

- Entropy của một biến ngẫu nhiên x đo tính không thể đoán trước của x

Intrinsic Info[6]

Intrinsic Info của một thuộc tính A được định nghĩa như sau:

$$IntrinsicInfo(S, A) = -\sum_{v \in Values(A)} \frac{|S_v|}{|S|} log(\frac{|S_v|}{|S|})$$

Với Values(A) là tập các giá trị của A và $S_v = \{x | x \in S, x_a = v\}$

Information Gain

Information gain của một thuộc tính A được định nghĩa như sau:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Với Values(A) là tập các giá trị của A và $S_v = \{x | x \in S, x_a = v\}$

- Ý nghĩa của Gain(S, A): lượng thông tin trung bình bị mất khi chia S theo A. Chúng ta cần chọn A sao cho Gain(S, A) là lớn nhất.

Information Gain Ratio

Information Gain Ratio của một thuộc tính A được định nghĩa như sau:

$$\text{GainRatio(S, A)} = \frac{Gain(S,A)}{IntrinsicInfo(S,A)}$$

GainRatio lớn khi dữ liệu được trải đều vào các lớp, ngược lại GainRatio nhỏ khi dữ liệu thuộc vào một lớp.

Thuật toán ID3

Thuật toán ID3 là thuật toán tham lam được đề xuất bởi Ross Quinlan năm 1986 và được dùng để học cây quyết định. Đây là thuật toán được thiết kế dưới dạng top-down.

Thuật toán hoạt động như sau:

+ Tại mỗi đỉnh N, chọn thuộc tính kiểm tra A để giúp chúng ta phân loại tốt nhất dữ liệu tại N.

+ Làm tương tự với các đỉnh còn lại đến khi nào cây phán đoán chính xác tất cả các dữ liệu huấn luyện hoặc các thuộc tính đã dùng hết rồi.

+ Mỗi thuộc tính chỉ được sử dụng nhiều nhất 1 lần trên đường đi từ gốc tới lá.

ID3 sử dụng Information Gain và Entropy để chọn thuộc tính kiểm tra A.

Thuật toán C4.5

Thuật toán C4.5 được phát triển dựa trên thuật toán ID3. Thuật toán có cách thức hoạt động tương tự thuật toán ID3, tuy nhiên nó sử dụng Information Gain Ratio để chọn thuộc tính kiểm tra. Hơn nữa, thuật toán có những cải tiến sau:

+ Thuật toán xử lý cả dữ liệu liên tục và rời rạc. Để xử lý dữ liệu liên tục, thuật toán C4.5 đặt ra một ngưỡng và chia thành 2 nhánh với 1 nhánh là các giá trị nhỏ hơn hoặc bằng ngưỡng và một nhánh là các giá trị lớn hơn ngưỡng

+ Thuật toán xử lý dữ liệu huấn luyện bị mất thuộc tính (bằng cách không xét dữ liệu đó trong quá trình tính toán entropy và information gain tại điểm đó)

+ Xử lý các thuộc tính với các giá trị khác nhau

+ Thuật toán có cắt tỉa sau khi tạo ra nhằm loại bỏ các cành không có tác dụng và thay thế chúng bởi các nút lá

Thuật toán C4.5 có tốc độ xử lý nhanh hơn và có độ chính xác cao hơn so với ID3. [6]

[7]

3.2.2 Random Forest

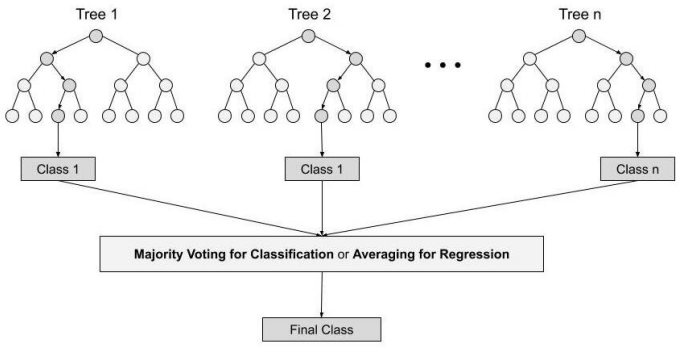
Định nghĩa

Random forest (RF) là một phương pháp của Leo Breiman (2001) cho cả 2 bài toán Classification và Regression.

Ý tưởng chính: dự đoán dựa trên sự kết hợp của nhiều cây quyết định, bằng cách lấy giá trị trung bình của tất cả các dự đoán riêng lẻ.

Mỗi cây trong rừng đơn giản nhưng ngẫu nhiên và phát triển theo một cách cách khác khác nhau theo cá thuộc tính được chọn và dữ liệu huấn luyện

RF hiện là một trong những phương pháp phổ biến và chính xác nhất. Nó có thể triển khai dễ dàng và hiệu quả. Đặc biệt nó có thể làm việc với số chiều rất cao mà không bị overfitting.



Hình 7: Random Forest.

Random Forest cho bài toán Classification

Đầu vào: Tập dữ liệu huấn luyện D

Quá trình học: Khởi tạo số lượng cây cần học. Với mỗi cây được học từ một tập dữ liệu ngẫu nhiên D_i có trùng lặp được chọn từ tập D. Tại mỗi node, chọn ngẫu nhiên một tập thuộc tính S và phát triển cây theo thuộc tính S tới kích thước tối đa mà không có cắt tỉa.

Dự đoán: Dự đoán bằng cách lấy trung bình tất cả các dự đoán từ mỗi cây độc lập

3.3 Naive Bayes

Naive Bayes là một tập hợp các thuật toán học có giám sát dựa trên việc áp dụng định lý Bayes với giả định "ngây thơ" về sự độc lập có điều kiện giữa các đặc trưng. Định lý Bayes

phát biểu mối quan hệ sau, cho mục tiêu y và vector đặc trưng X:

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n|y)}{P(x_1, ..., x_n)}$$

Với các giả thiết độc lập có điều kiện được giả định:

$$P(x_i|y, x_1, ..., x_{i-1}, x_{i+1}, ..., x_n) = P(x_i|y),$$

do đó, với tất cả i, mối quan hệ có thể đơn giản hoá thành

$$P(y|x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, .., x_n)}$$

Vì $P(x_1, ..., x_n)$ không đổi trong đầu vào, nên có thể sử dụng quy tắc phân loại sau:

$$\begin{aligned} P(y \mid x_1, \ldots, x_n) &\propto P(y) \prod_{i=1}^{n} P(x_i \mid y) \\ &\Downarrow \\ \hat{y} &= \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y), \end{aligned} \tag{1}$$

và chúng ta có thể sử dùn Maximum A Posteriori (MAP) để ước lượng $P(y)$, $P(x_i|y)$, và tần số xuất hiện tương đối của lớp y trong tập huấn luyện.

Sự khác nhau của các mô hình phân loại navie Bayes chủ yếu dựa trên các giả định liên quan đến xác suất của $P(x_i|y)$. Mặc dù những giả định được đơn giản hóa quá mức, nhưng phân loại naive Bayes vẫn hoạt động khá tốt trong nhiều tình huống thực tế, điển hình là phân loại tài liệu và lọc thư rác. Thuật toán chỉ yêu cầu một lượng nhỏ dữ liệu huấn luyện để ước tính các thông số cần thiết.

Một số thuật toán phân loại Naive Bayes thường dùng:

Gaussian Naive Bayes

GaussianNB triển khai thuật toán Gaussian Naive Bayes để phân loại. Trong đó, các đặc trưng tuân thủ theo phân phối Gaussian:

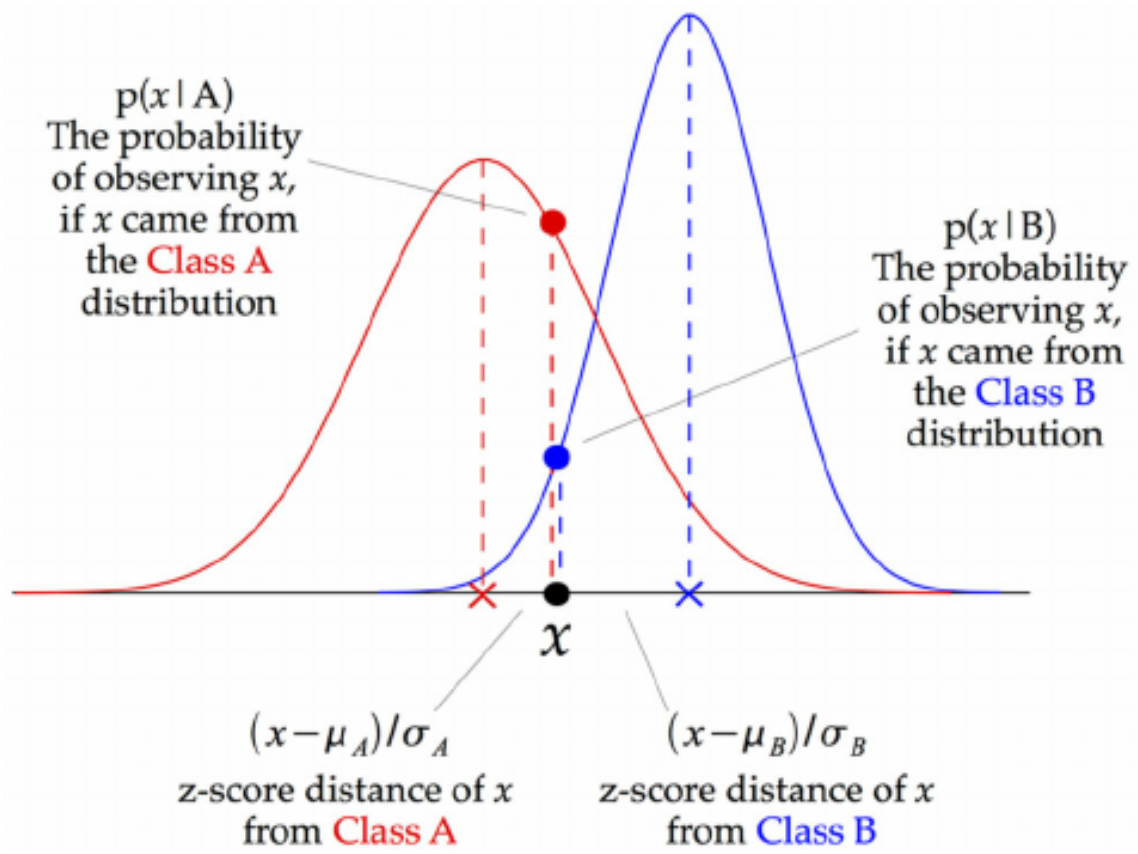
$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

trong đó, các tham số σ_y và μ_y được ước lượng bằng maximum likelihood:

$$(\mu_y, \sigma_y^2) = \arg\max_{\mu_y, \sigma_y^2} \prod_{n=1}^{N} p(x_i^{(n)} | \mu_y, \sigma_y^2)$$

Mô hình này được sử dụng chủ yếu trong loại dữ liệu mà các thành phần là các biến liên tục.

Một cách tiếp cận để tạo một mô hình đơn giản là giả định rằng dữ liệu được mô tả bằng phân phối Gauss không có đồng phương sai giữa các chiều. Mô hình này có thể phù hợp bằng cách đơn giản tìm giá trị trung bình và độ lệch chuẩn của các điểm trong mỗi nhãn, đó là tất cả những gì cần thiết để xác định một phân phối như vậy. (Hình 8)



Hình 8: Mô hình Gaussian naive Bayes đơn giản. [8]

Hình minh hoạ trên chỉ ra cách mà một mô hình Gaussian Naive Bayes phân loại hoạt động. Tại mỗi điểm dữ liệu, mô hình sẽ dễ dàng tính được xác suất xảy ra của quan sát x tương ứng với từng lớp, từ đó có thể đưa ra dự đoán.

Như vậy, mô hình Gaussian Naive Bayes hoạt động dựa trên giả thiết các quan sát là độc lập và tuân theo phân phối Gauss. Mô hình nhìn có vẻ "ngây thơ" nhưng lại hoạt động rất tốt trong một số trường hợp.

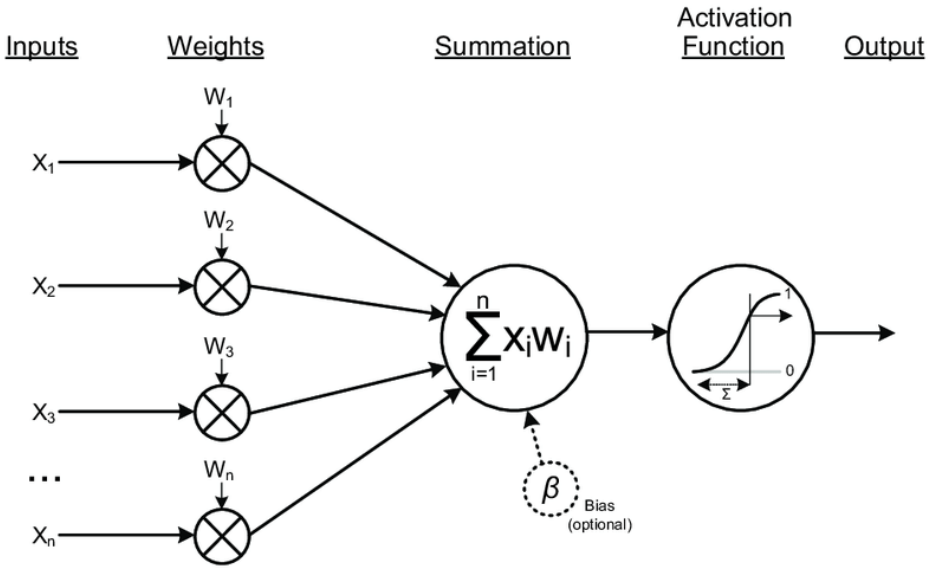
3.4 Artificial Neural Network

Mạng neuron nhân tạo (Artificial Neural Network) mô phỏng hệ thống neuron sinh học trong bộ não của con người. Mô hình này sẽ đi xấp xỉ một hàm dựa vào một số mẫu giá trị vào và ra của hàm đó. Đây là cấu trúc xử lý thông tin xong xong và phi tập trung (do đó có thể tận dụng bộ xử lý các tác vụ có liên quan tới đồ họa *GPU* để tăng tốc độ tính toán). Mô hình này có khả năng học, ghi nhớ và vận dụng lại các tri thức đã học được; nhưng chất lượng phụ thuộc và các yếu tố sau:

- Kiến trúc mạng.
- Đặc điểm của các dữ liêu đầu vào và đầu ra.
- Thuật toán học.
- Dữ liệu huấn luyện.

Tín hiệu vào của mỗi một neuron (mỗi nốt trong mạng), trừ những nốt đầu tiên, là các nốt x_i, với $i = \overline{1, m}$. Ứng với mỗi nốt này là một trọng số w_i, $i = \overline{1, m}$. Ngoài ra còn có trọng số w_0. *Net input* được định nghĩ là một hàm dạng tuyến tính như sau:

$$Net(\boldsymbol{w}, \boldsymbol{x}) = w_0 + \sum_{i=1}^{m} x_i w_i$$

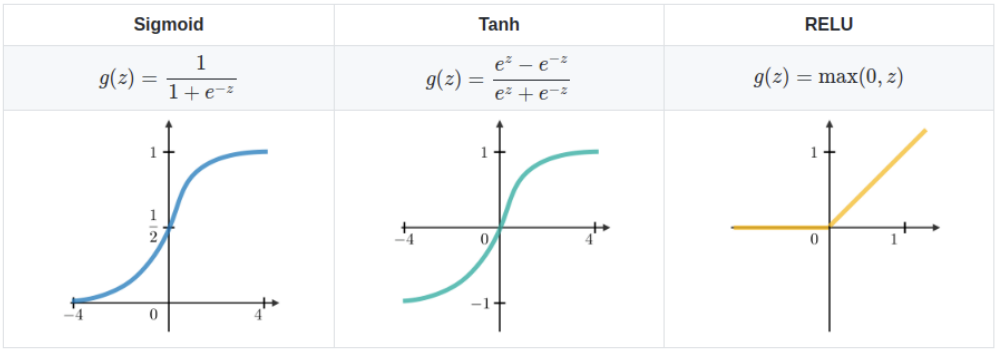


Hình 9: Minh họa tín hiệu vào và ra đến một neuron [9].

Tuy nhiên, *Net input* còn cần được đưa qua một hàm kích hoạt (activation function) $f(\cdot)$ để tạo thành giá trị đầu ra (Hình 9):

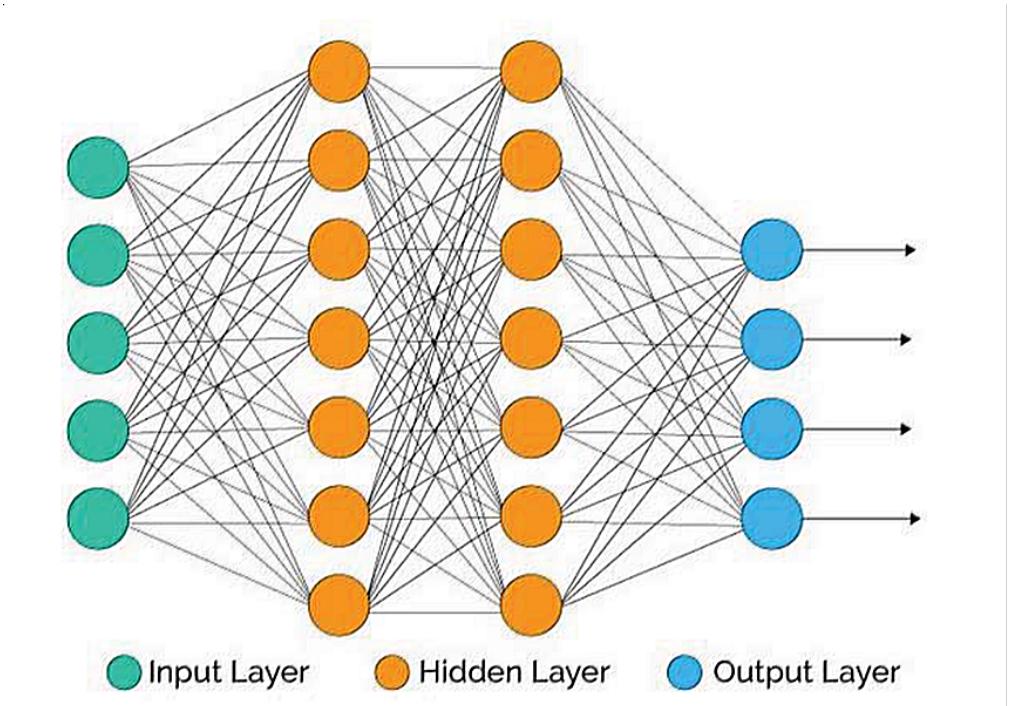
$$Out(\boldsymbol{w},\ \boldsymbol{x}) = f(Net(\boldsymbol{w},\ \boldsymbol{x}))$$

Ba hàm kích hoạt phổ biến là Sigmoid, Tanh, và ReLU (Hình 10).



Hình 10: Ba hàm kích hoạt phổ biến [10].

Mỗi lớp của mạng neuron sẽ gồm nhiều lớp. "Lớp trung gian" hay "lớp ẩn" là các lớp ở giữa "lớp đầu vào" và "lớp đầu ra" (Hình 11). Lớp đầu vào và lớp đầu ra luôn cần phải có, còn các lớp trung gian có hay không cũng được và không bị giới hạn về số lượng. Một mạng neuron được gọi là kết nối đầy đủ (fully connected) nếu các đầu ra của một lớp kết nối với tất cả các neuron của lớp tiếp theo. Cấu trúc của một mạng neuron nhân tạo được thể hiện qua 5 đặc tính:



Hình 11: Các lớp trong mạng neuron nhân tạo [11].

- Số đặc tính của tín hiệu vào và ra.
- Số lớp.
- Số neuron (số nút) ở mỗi lớp.
- Số lượng neuron kết nối với một neuron khác.
- Sự kết nối của các neuron trong cùng lớp hoặc giữa các lớp với nhau.

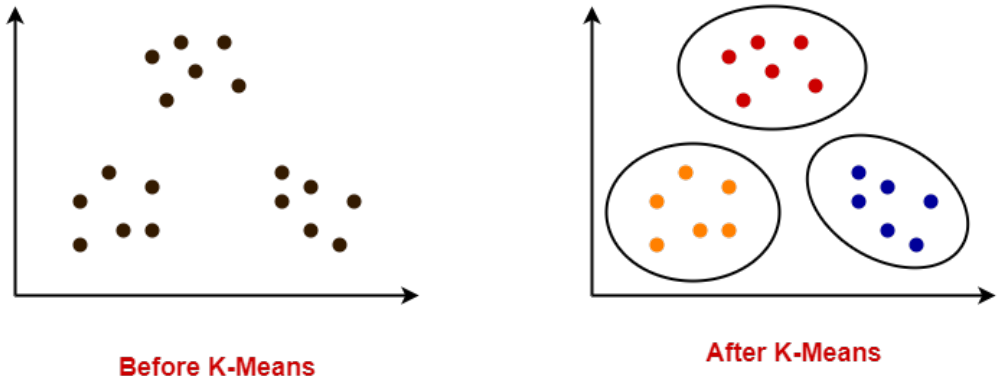
Mô hình trong báo cáo sẽ được huấn luyện theo hướng điều chỉnh các tham số trong mạng, dựa trên việc tối thiểu hàm lỗi:

$$L(\mathbf{w}) = \frac{1}{|\mathbf{D}|} \sum_{x \in \mathbf{D}} loss(d_x, \text{out}(x))$$

Trong đó, $\mathbf{D}$ là tập dữ liệu huấn luyện, mỗi đầu vào x có nhãn tương ứng là d_x. Mỗi lần sẽ lấy một số lượng mẫu để huấn luyện mô hình (batch size). Một lần đi qua toàn bộ mẫu huấn luyện được gọi là một *epoch*.

3.5 K-means Clustering

K-means Clustering là mô hình học máy *không giám sát*. Mô hình hướng đến việc phân chia tập dữ liệu chưa biết nhãn thành các cụm (Hình 12). Để áp dụng vào bài toán đặt ra, ban đầu mô hình sẽ được sử dụng để phân chia các mẫu huấn luyện thành 3 cụm. Mục đích sử dụng mô hình này là để kiểm tra xem liệu đặc tính được xem xét đến của 3 loài chim cánh cụt có sự khác biệt quá nhiều không. Dựa trên số lượng ở ba cụm mà tạm gán nhãn cho từng nhóm, sau đó các mẫu kiêm tra sẽ được sử dụng để xem khả năng dự đoán của mô hình.



Hình 12: Minh họa một kết quả phân cụm của K-means Clustering [12].

Với số lượng cụm cho trước, luồng hoạt động của thuật toán như sau:

1. Chọn vị trí bất kỳ cho tâm các cụm.
2. Phân mỗi điểm dữ liệu vào cụm có tâm "gần" nó nhất.
3. Kiểm tra thuật toán đã kết thúc theo luật đã định trước hay chưa.
4. Cập nhật lại tâm các cụm
5. Quay lại bước 2.

Ở bước 1, có thể khởi tạo các tâm như sau (k-means++):

- Chọn ngẫu nhiên tâm đầu tiên.
- Chọn tâm thứ hai "xa" nhất so với tâm thứ nhất.
- ...
- Chọn tâm thứ i "xa" nhất so với các tâm đã khởi tạo.
- ...

Có thể tự định nghĩa ra hàm đo khoảng cách, nhưng phổ biến nhất là đo theo khoảng cách Euclid:

$$d(\mathbf{x}, \mathbf{c}) = ||\mathbf{x} - \mathbf{c}|| = \sqrt{\sum_{i=1}^{n}(x_i - c_i)^2}$$

trong đó $\mathbf{x}$ và $\mathbf{c}$ là các vector lần lượt ứng với một điểm dữ liệu và một tâm. Các điều kiện phổ biến có thể sử dụng để dừng thuật toán là:

- Số lượng mẫu được phân sang cụm hết nhỏ hơn một ngưỡng nào đó.
- Vị trí các tâm dịch chuyển ít hơn một ngưỡng nào đó.
- Tổng (lỗi) $\sum_{i=1}^{k}\sum_{x\in C_i} d(\mathbf{x}, \mathbf{m_i})^2$ thay đổi ít hơn một ngướng nào đó. k là số cụm, $\mathbf{m_i}$ là tâm của cụm (tập các điểm cùng nhóm) C_i.

Các tâm có thể cập nhật theo công thức sau:

$$\mathbf{m_i} = \frac{1}{C_i}\sum_{x\in C_i} x$$

4 Kết quả thực nghiệm

Phần này của báo cáo sẽ tiến hành thực nghiệm để kiểm tra kết quả dự đoán của năm mô hình học máy khác nhau. Đầu tiên, bốn mô hình học máy có giám sát sẽ được thay đổi tham số, huấn luyện trên các tập dữ liệu có kích thước khác nhau và quan sát độ chính xác đạt được trên bộ dữ liệu xác thực. Sau khi lựa chọn được các tham số có kết quả tốt nhất ở quá trình thực nghiệm trước đó, các thuật toán có giám sát sẽ được đo độ chính xác của khả năng dự đoán trên bộ dữ liệu kiểm tra. Ngoài ra, thuật toán K-means Clustering cũng sẽ được sử dụng với cách tiếp cận theo hướng không giám sát, để kiểm tra khả năng dự đoán so với bốn thuật toán có giám sát. Ngôn ngữ lập trình sử dụng là Python, framework Scikit-learn và thư viện Keras được sử dụng để xây dựng các mô hình học máy.

4.1 K-nearest Neighbor

Với KNN, mô hình sẽ được kiểm tra độ hiệu quả đạt được khi thay đổi số lượng hàng xóm và trọng số khoảng cách giữa các hàng xóm. Mô hình sử dụng thư viện KNeighborsClassifier trong gói sklearn.neighbors để phân loại. Mô hình KNN sẽ thử nghiệm với các hàng xóm bằng : 1, 3, 5 , kết hợp với 3 hàm tính trọng số khoảng cách , để đánh giá mức độ hiệu quả của mô hình. Các hàm được sử dụng để gán trọng số khoảng cách là:

$$Weight1 = e^{-\frac{distances^2}{\sigma^2}}$$

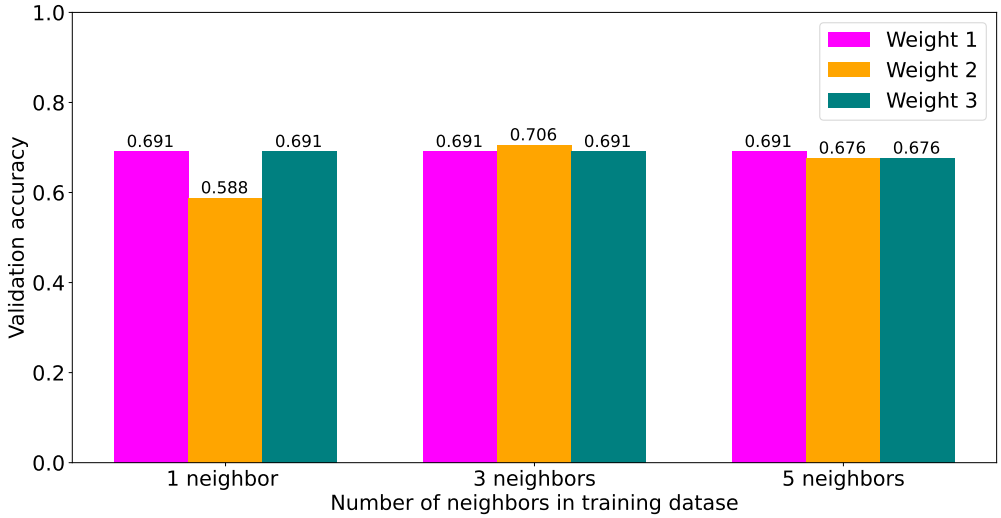
$$Weight2 = \frac{1}{distances}$$

$$Weight3 = \frac{1}{distances^2}$$

trong đó, *distances* là khoảng cách giữa 2 điểm dữ liệu.

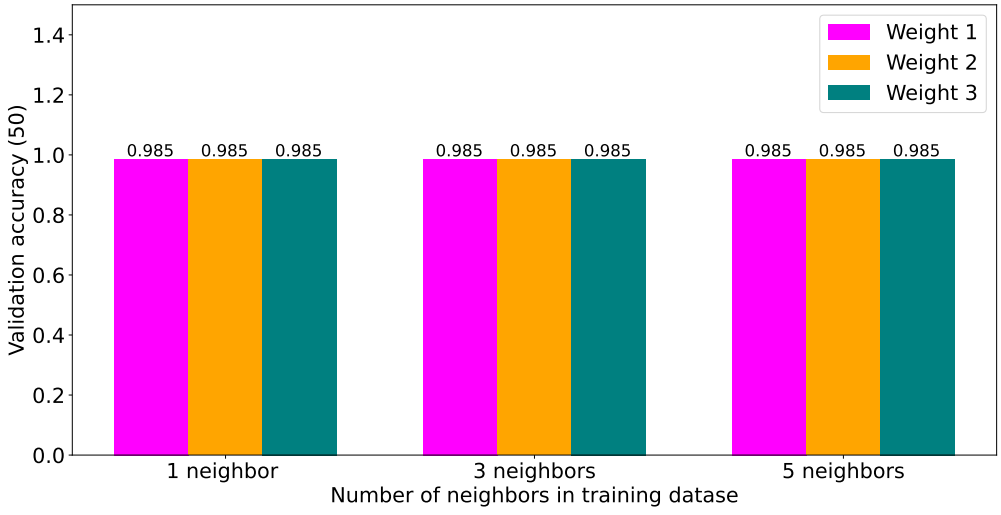
Tương ứng với mỗi tham số trên, mô hình sẽ lần lượt được huấn luyện với các bộ dữ liệu gồm: 50, 110, và 170 mẫu. Mô hình sẽ thử lần lượt trên các bộ dữ liệu chưa được chuẩn hoá và đã được chuẩn hoá.

Độ hiệu quả của mô hình KNN trên tập xác thực với bộ dữ liệu huấn luyện gồm 110 mẫu chưa được chuẩn hoá, kết hợp với các tham số như: số hàng xóm, cách tính trọng số,.. được thể hiện trong hình 13.

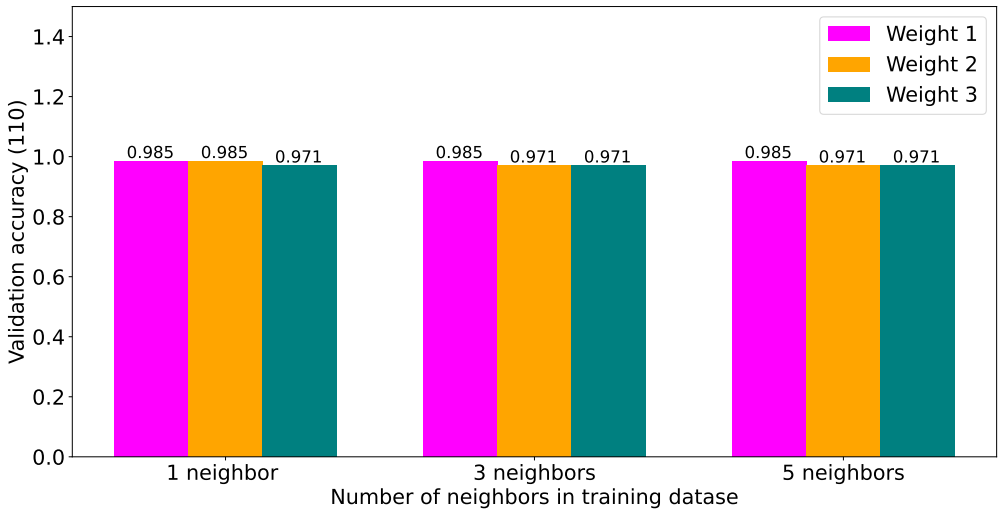


Hình 13: Độ chính xác trên tập xác thực ứng với số lượng mẫu huấn luyện là 110 khi sử dụng số hàng xóm và trọng số khoảng cách khác nhau.

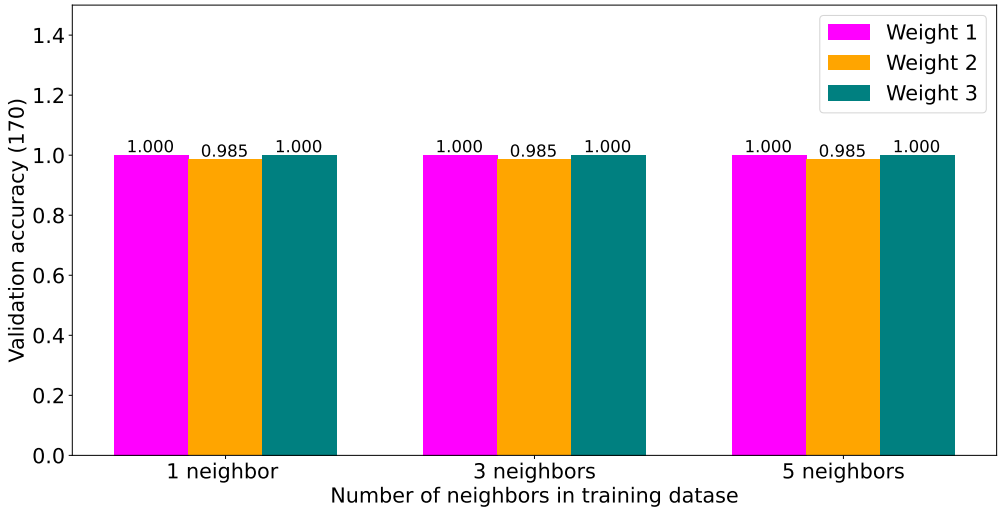
Với bộ dữ liệu gồm $50, 110, 170$ mẫu đã được chuẩn hoá, mô hình KNN cho hiệu quả trên tập xác thực được thể hiện trong lần lượt các hình 14, 15, 16. Có thể thấy rõ, với tập dữ liệu huấn luyện chưa được chuẩn hoá, kết quả thu được sẽ kém hơn với bộ dữ liệu đã được chuẩn hoá. Chứng tỏ, trong tập dữ liệu Chim cánh cụt có thể tồn tại một số những đặc trưng sẽ lấn át những đặc trưng còn lại, dẫn đến kết quả dự đoán bị ảnh hưởng. Và việc chuẩn hoá dữ liệu đã giúp cân bằng lại sự ảnh hưởng giữa các thuộc tính, giúp cho việc dự đoán trở nên chính xác hơn.



Hình 14: Độ chính xác trên tập xác thực ứng với số lượng mẫu huấn luyện là 50 trên tập dữ liệu chuẩn hoá khi sử dụng số hàng xóm và trọng số khoảng cách khác nhau.



Hình 15: Độ chính xác trên tập xác thực ứng với số lượng mẫu huấn luyện là 110 trên tập dữ liệu chuẩn hoá khi sử dụng số hàng xóm và trọng số khoảng cách khác nhau.



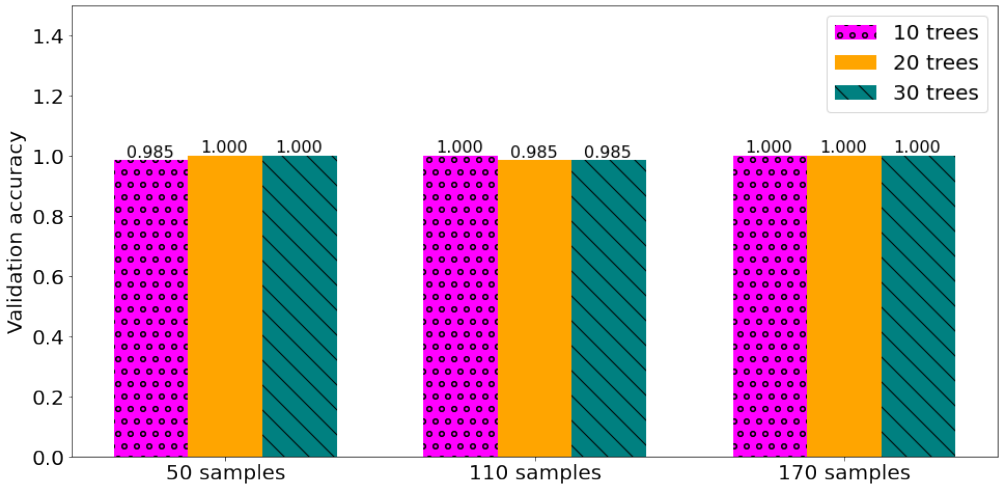
Hình 16: Độ chính xác trên tập xác thực ứng với số lượng mẫu huấn luyện là 170 trên tập dữ liệu chuẩn hoá khi sử dụng số hàng xóm và trọng số khoảng cách khác nhau.

4.2 Random Forest

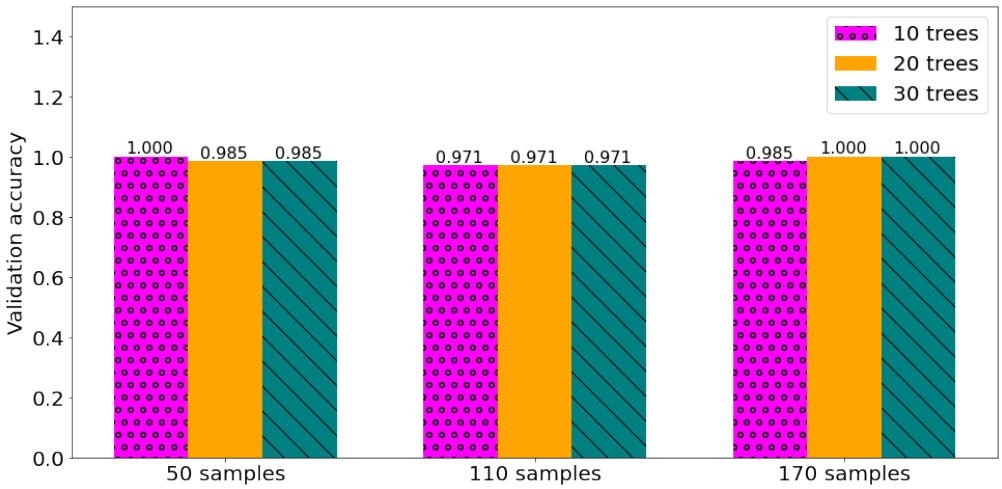
Mô hình Random Forest để giải bài toán phân loại chim cánh cụt sử dụng thuật toán C4.5 để tạo cây phân loại và sử dụng hàm đánh giá 'entropy' để đánh giá thuộc tính và chia nhánh dữ liệu [13]. Trong phần này, mô hình sẽ được kiểm tra độ hiệu quả đạt được khi thay đổi số lượng cây quyết định tạo nên một rừng và số lượng các thuộc tính được sử dụng để dùng hàm entropy đánh giá. Với dữ liệu sử dụng trong mô hình này, do mô hình áp dụng hàm enttropy để phân loại, nên thông qua công thức của hàm entropy ta thấy rằng với cùng một mô hình thì dữ liệu được chuẩn hóa hay không chuẩn hóa sẽ cho ra cùng kết quả như

nhau.

Trong quá trình thực nghiệm chúng tôi thấy rằng với số thuộc tính được chọn để đánh giá và chia nhánh dữ liệu có sự ngẫu nhiên cao sẽ cho kết quả tốt hơn với các thuộc tính cho sự ngẫu nhiên thấp. Một ví dụ được thể hiện ở hình 17 và 18 dưới đây. Với số lượng thuộc tính của bộ dữ liệu chúng tôi dùng để phân loại (7 thuộc tính) thì số lượng thuộc tính được chọn để đánh giá và phân nhân tốt nhất là $2 = \sqrt{max\_features}$ (max_features là số thuộc tính tối đa).



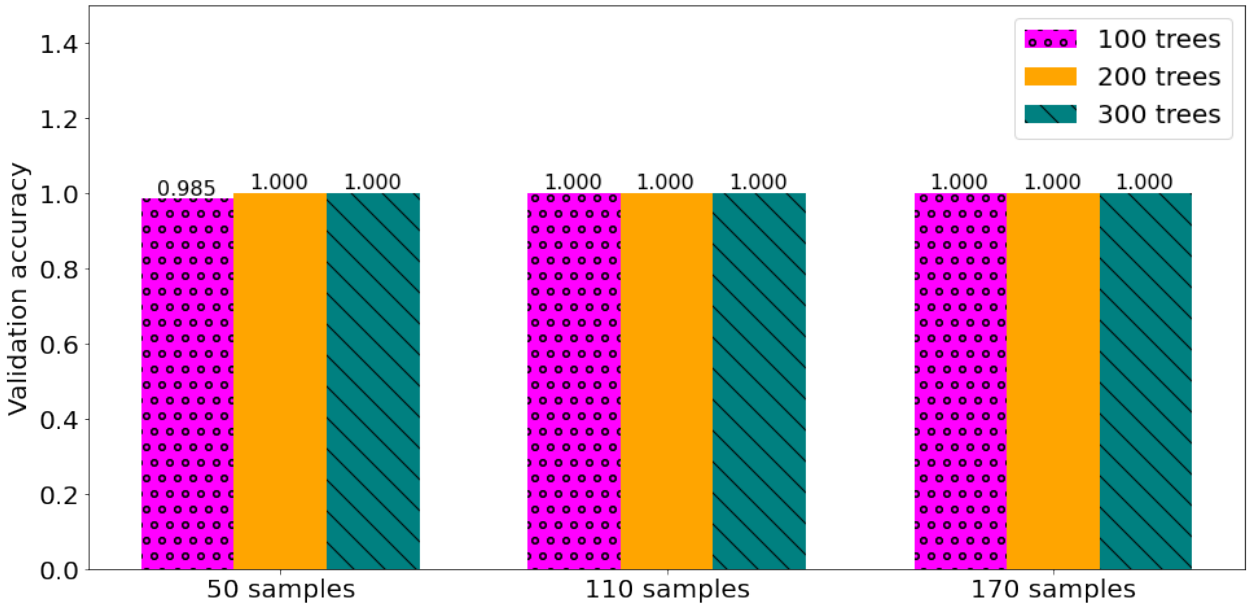
Hình 17: Độ chính xác trên tập xác thực ứng với số lượng mẫu huấn luyện khác nhau khi tăng dần số lượng cây quyết định, sử dụng $\sqrt{max\_features}$ thuộc tính để chia nhánh dữ liệu với số lượng cây nhỏ.



Hình 18: Độ chính xác trên tập xác thực ứng với số lượng mẫu huấn luyện khác nhau khi tăng dần số lượng cây quyết định, sử dụng $max\_features$ thuộc tính để chia nhánh dữ liệu với số lượng cây nhỏ.

Hình 17 cho ta thấy độ chính xác trên tập xác thực ứng với số lượng mẫu huấn luyện

khác nhau khi tăng dần số cây quyết định, sử dụng $\sqrt{max\_features}$ thuộc tính để chia nhánh dữ liệu. Ta thấy rằng với số lượng cây nhỏ, khi tăng dần số lượng nhỏ cây quyết định, độ chính xác trên tập xác thực không hoàn toàn phụ thuộc vào số lượng cây do mỗi cây quyết định có yếu tố ngẫu nhiên cao. Tuy nhiên độ chính xác trên tập xác thực luôn trên mức 95% và có thể đạt tới mức 100%. Bên cạnh đấy, với số lượng mẫu huấn luyện tăng lên đến 170 mẫu luôn cho kết quả tốt hơn so với số mẫu huấn luyện còn lại, điều này cho thấy với số lượng mẫu huấn luyện đủ lớn thì sẽ cho độ chính xác tốt hơn.

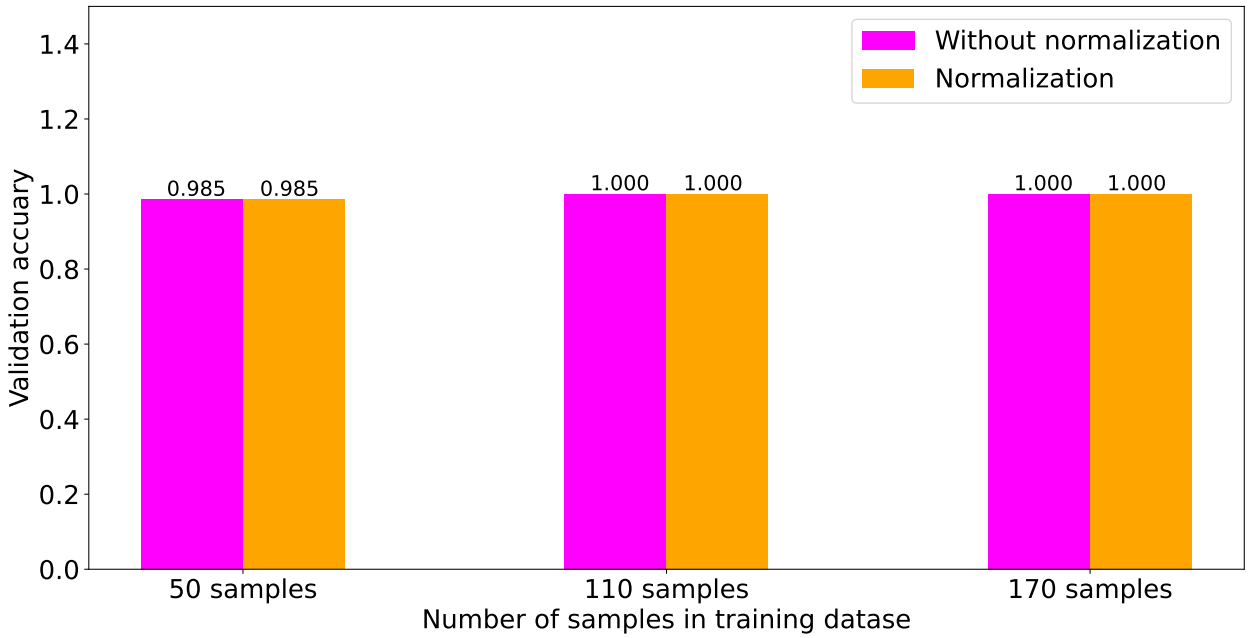


Hình 19: Độ chính xác trên tập xác thực ứng với số lượng mẫu huấn luyện khác nhau khi tăng dần số lượng cây quyết định, sử dụng $\sqrt{max\_features}$ thuộc tính để chia nhánh dữ liệu với số lượng cây lớn hơn 100.

4.3 Naive Bayes

Trong phần thử nghiệm mô hình Naive Bayes, thuật toán Gaussian Navie Bayes (GNB) sẽ được áp dụng để phân loại. Mô hình GNB sẽ giả định các đặc trưng của những con chim cánh cụt là độc lập với nhau.

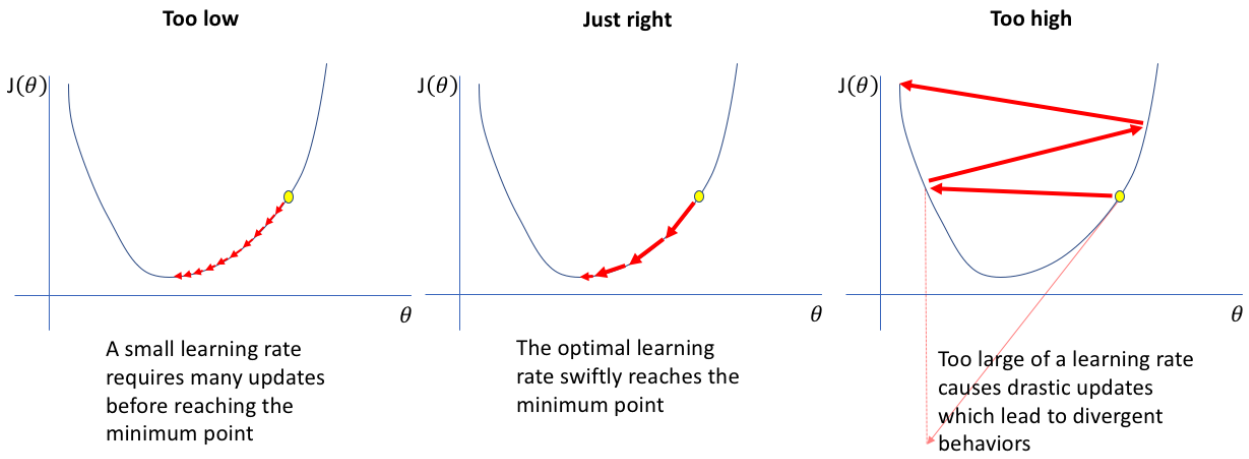
Hình 20 mô tả độ hiệu quả của mô hình khi huấn luyện trên các tập dữ liệu gồm 50, 110, 170 mẫu, tương ứng với hai bộ dữ liệu chưa chuẩn hoá và chuẩn hoá. Trong trường hợp này, mô hình GNB tuy đơn giản nhưng lại hoạt động hiệu quả. Với tập dữ liệu gồm 50, 110, 170 mẫu, và trên tập dữ liệu chưa được chuẩn hoá, cũng như được chuẩn hoá, thì mô hình đều đưa ra kết quả đúng gần như 100%. Dữ liệu chuẩn hóa hay không cũng không quan trọng vì mô hình này học dựa trên phấn phối các giá trị của từng đạc tính. Độ chính xác cao phần nào chỉ ra các đặc trưng của dữ liệu có mối tương quan tương đối thấp.



Hình 20: Độ chính xác trên tập xác thực ứng với số lượng mẫu huấn luyện khác nhau trên tập dữ liệu chưa chuẩn hoá và tập dữ liệu chuẩn hoá.

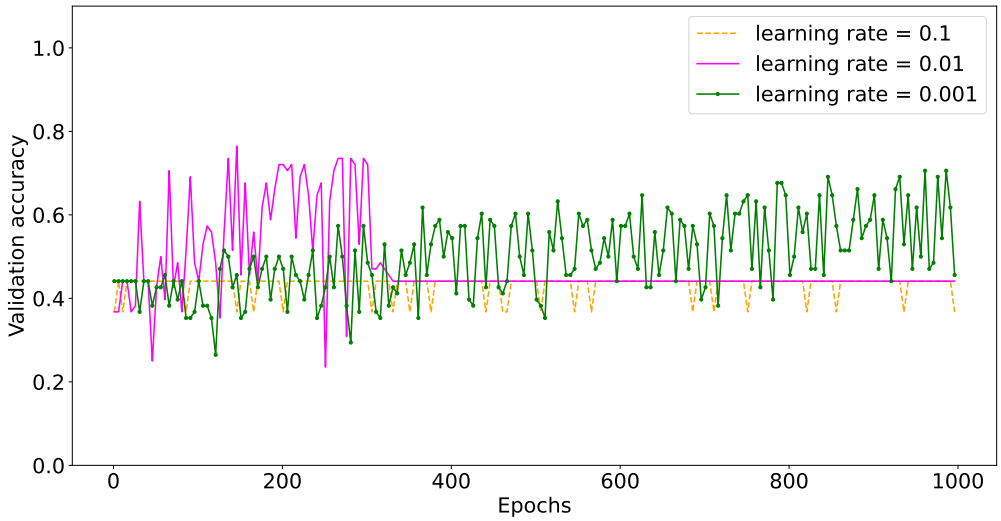
4.4 Artificial Neural Network

Trong phần này, mô hình mạng neuron nhân tạo (Artificial Neural Network - ANN) sẽ được kiểm tra độ hiệu quả đạt được khi thay đổi các tham số liên quan đến tốc độ học (learning rate) và hàm kích hoạt (activation function). Mô hình tuần tự (Sequence) sẽ được xây dựng với hai lớp trung gian kết nối đầy đủ (*dense layer* hay *fully-connected layer*, nghĩa là mỗi neuron của từng lớp sẽ nhận đầu vào của tất cả các neuron lớp trước đó. Mỗi lớp trung gian sẽ có 16 nút. Lớp đầu vào sẽ có 7 nút ứng với từng thuộc tính còn lớp đầu ra sẽ có 3 nút ứng với từng lớp phân loại Hàm kích hoạt của lớp đầu ra là *Softmax*, phù hợp với bài toán phân loại vì nó cho biết mức độ tự tin khi phân một mẫu cụ thể vào từng lớp. Thuật toán tối ưu (optimizer) được sử dụng sẽ là *Adam*. Hàm lỗi là *Categorical Crossentropy*. Kích thước của batch, hay số mẫu cho một lần huấn luyện là 16.



Hình 21: Minh họa ảnh hưởng của tốc độ học đến chất lượng của mô hình [14].

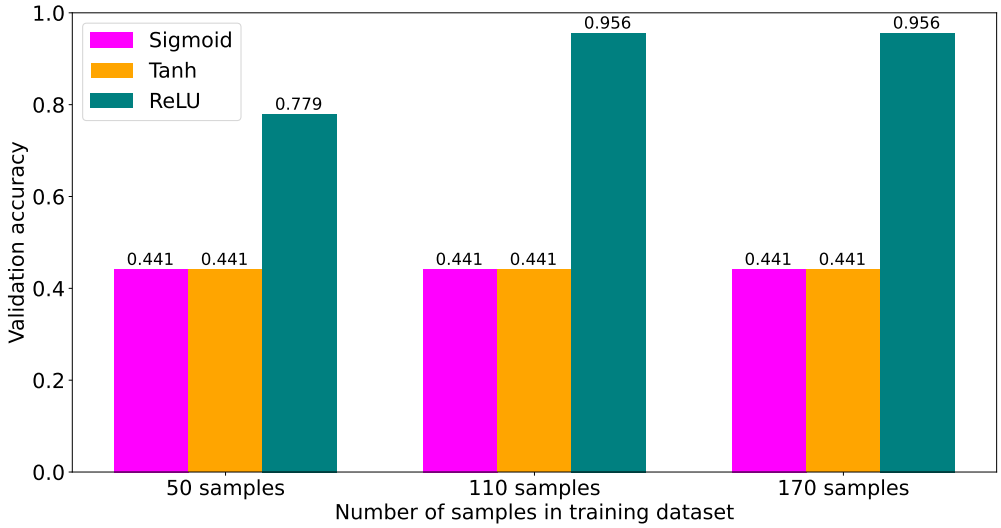
Hình 22 kiểm tra độ chính xác khi dự đoán trên tập xác thực của mô hình khi huấn luyện với các tốc độ học khác nhau (0.1, 0.01 và 0.001). Hàm kích hoạt của các lớp trung gian đang được cố định là *ReLU* (Rectified Linear Unit). Có thể thấy, khi tốc độ học quá lớn, độ chính xác khá thấp, cụ thể là không vượt quá ngưỡng 44.12 %. Trong trường hợp này, mô hình sẽ có sự học không ổn định vì bị ảnh hưởng quá nhiều vào giá trị theo hướng giảm gradient. Nói cách khác, vì "bước nhảy" quá lớn, thuật toán sẽ khó có thể chạm tới điểm tối ưu (Loanh quanh gần điểm tối ưu như trường hợp bên phải của Hình 21), hoặc đôi khi nhảy qua cả điểm tối ưu cần đạt tới và dễ rơi vào cực tiểu địa phương. Khi mà tốc độ học quá nhỏ (0.001), việc học có vẻ rất ổn định nhưng lại rất lâu, khi chạy 1000 epoch, độ chính xác đạt được cao nhất là 72.06 % tại epoch thứ 902. Trong khi đó, nếu tốc độ học là



Hình 22: Độ chính xác trên bộ dữ liệu xác thực qua từng epoch khi tốt độ học là 0.1, 0.01, 0.001.

0.01, độ chính xác cao nhất là 77.94 % tại epoch thứ 194, tức là nhanh hơn rất nhiều. Tuy nhiên, khoảng từ epoch thứ 380 trở đi, mô hình bị *overfitting* dẫn đến độ chính xác trên bộ dữ liệu xác thực rất thấp.

Độ chính xác cao nhất đạt được trên bộ dữ liệu xác thực khi áp dụng các hàm kích hoạt khác nhau được biểu diễn trong Hình 23. Điều dễ thấy nhất là hàm ReLU đang có kết quả cao vượt trội so với Sigmoid và Tanh. Hàm Sigmoid có điểm yếu là dễ bị bão hòa và bị ảnh hưởng bởi sự suy giảm gradient. Khi đầu vào có trị tuyệt đối lớn (rất âm hoặc rất dương), gradient của hàm số này sẽ rất gần với 0. Điều này đồng nghĩa với việc các hệ số tương ứng với các nốt đang xét sẽ gần như không được cập nhật (còn được gọi là *vanishing gradient*). Vấn đề này có thể giải quyết phần nào nếu dữ liệu được chuẩn hóa về dạng có giá trị kỳ vọng bằng 0. Hàm Tanh cũng dễ bị bão hòa về hai đầu, nhưng hàm này có giá trị gradient nói chung lớn hơn so với Sigmoid và nếu giá trị của các thuộc tính đầu vào đối xứng xung quanh điểm không, tốc độ hội tụ sẽ vô cùng nhanh. Do đó, để cải thiện chất lượng của hai hàm kích hoạt này, dữ liệu cần được chuẩn hóa.



Hình 23: Độ chính xác trên tập xác thực ứng với số lượng mẫu huấn luyện khác nhau khi sử dụng các hàm kích hoạt Sigmoid, Tanh, và ReLU.

Bảng 2 biểu diễn độ chính xác đạt được và epoch sớm nhất để đạt được độ chính xác cao nhất khi sử dụng các hàm kích hoạt khác nhau trên bộ dữ liệu đã chuẩn hóa. Giữa các lớp của kiến trúc mạng cũng được thêm một lớp *batch normalization* với khả năng chuẩn hóa đầu ra của lớp trước về dạng có kỳ vọng 0 và độ lệch chuẩn 1. Số epoch tối đa là 100. Có thể thấy trong trường hợp chỉ có 50 mẫu huấn luyện, Tanh là hàm kích hoạt duy nhất mang lại độ chính xác 100 %. Sigmoid có độ chính xác cao hơn ReLU nhưng mất tới tận epoch thứ 98 mới có được kết quả này so với chỉ 13 epoch của Tanh và 19 epoch của ReLU. Với các

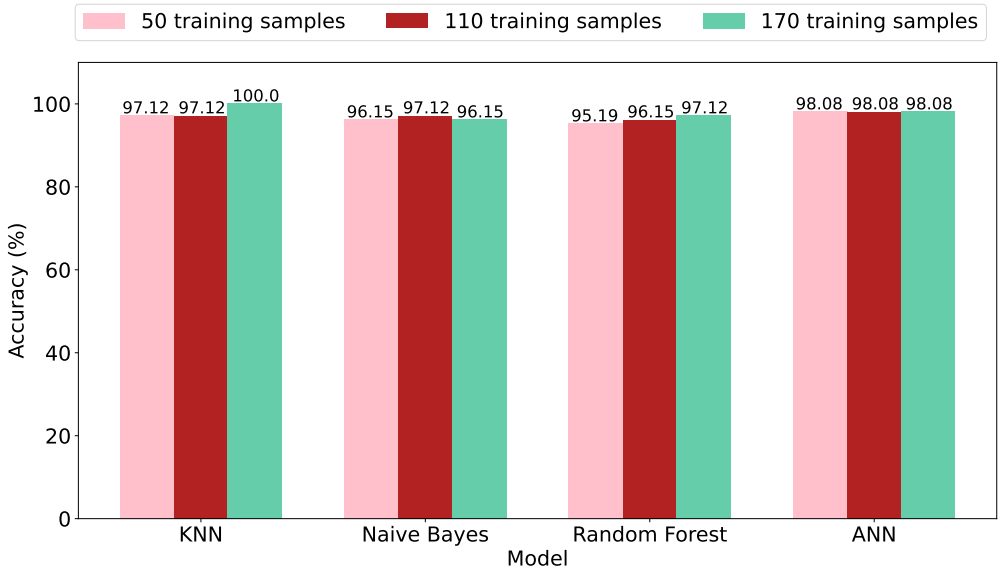
bộ dữ liệu huấn luyện lớn hơn, cả ba hàm kích hoạt đều có độ chính xác tuyệt đối. Trong đó hàm Tanh có số epoch tối thiểu để đạt kết quả cao nhất trong 2/3 bộ dữ liệu và không bị kém nhất ở bộ dữ liệu nào. Kết quả của Tanh rất cao và ổn định, khi số mẫu huấn luyện càng nhiều thì số epoch yêu cầu càng ít hơn.

Kích hoạt	**50 mẫu huấn luyện**		**110 mẫu huấn luyện**		**170 mẫu huấn luyện**	
	Độ chính xác	Epoch	Độ chính xác	Epoch	Độ chính xác	Epoch
Sigmoid	98.53 %	98	100 %	3	100 %	13
Tanh	100 %	13	100 %	8	100 %	3
ReLU	97.05 %	20	100 %	9	100 %	4

Bảng 2: Độ chính xác trên tập xác thực ứng với dữ liệu đã chuẩn hóa khi sử dụng các hàm kích hoạt Sigmoid, Tanh, và ReLU.

4.5 So sánh kết quả các thuật toán có giám sát trên bộ dữ liệu kiểm tra

Sau khi lựa chọn được các tham số đem lại kết quả tốt nhất ở những bước thực nghiệm trước, các mô hình sẽ được đem dự đoán loài chim cánh cụt trên bộ dữ liệu kiểm tra. Độ chính xác được biểu diễn trong hình 24.



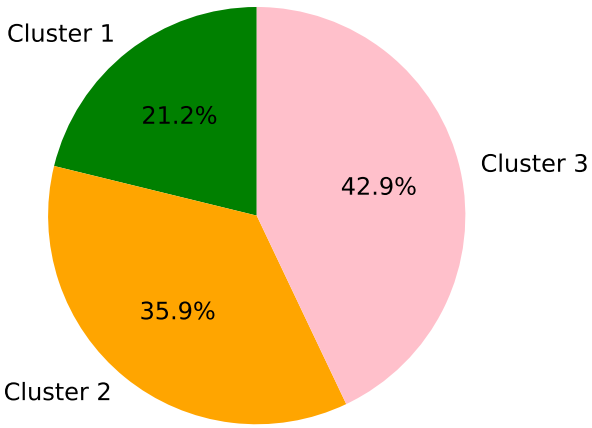
Hình 24: Độ chính xác trên tập kiểm tra ứng với mỗi mô hình khi sử dụng các bộ dữ liệu huấn luyện với số mẫu khác nhau.

Nhìn chung, khi số lượng mẫu huấn luyện là 50, cùng một mô hình luôn cho độ chính xác thấp nhất. Khi số lượng mẫu huấn luyện tăng thành 170, độ chính xác đạt được cao nhất với 3/4 mô hình sử dụng, ngoại trừ Naive Bayes. Có thể thấy, không phải lúc nào nhiều dữ liệu hơn cũng cho kết quả tốt hơn. Random Forest đang cho kết quả kém nhất ở cả hai trường hợp sử dụng bộ liệu huấn luyện gồm 50 và 110 mẫu. KNN là mô hình duy nhất đạt được độ chính xác là 100 % (khi huấn luyện trên 170 mẫu dữ liệu), mặc dù hai trường hợp còn lại có kết quả không quá nổi trội. Đáng chú ý là ANN có độ chính xác không đổi ở mức 98.08 %, kể cả khi huấn luyện chỉ với 50 mẫu, nghĩa là khả năng tổng hợp hóa của ANN là rất cao. Mô hình này đem lại kết quả cao nhất trong hai trên ba trường hợp, chỉ kém hơn KNN khi huấn luyện trên 170 mẫu dữ liệu.

Các mô hình học máy *có giám sát* đều cho độ chính xác rất cao. Khả năng là do các đặc trưng đang được quan sát của mỗi loài có sự phân biệt tương đối rõ ràng. Để kiểm chứng phần nào nhận xét này, tiếp theo, mô hình phân cụm sẽ được sử dụng để quan sát xem liệu kể cả khi không biết trước nhãn, mô hình có thể dễ dàng phân biệt được tốt cả ba loại chim dựa vào các đặc trưng quan sát được hay không.

4.6 K-means Clustering

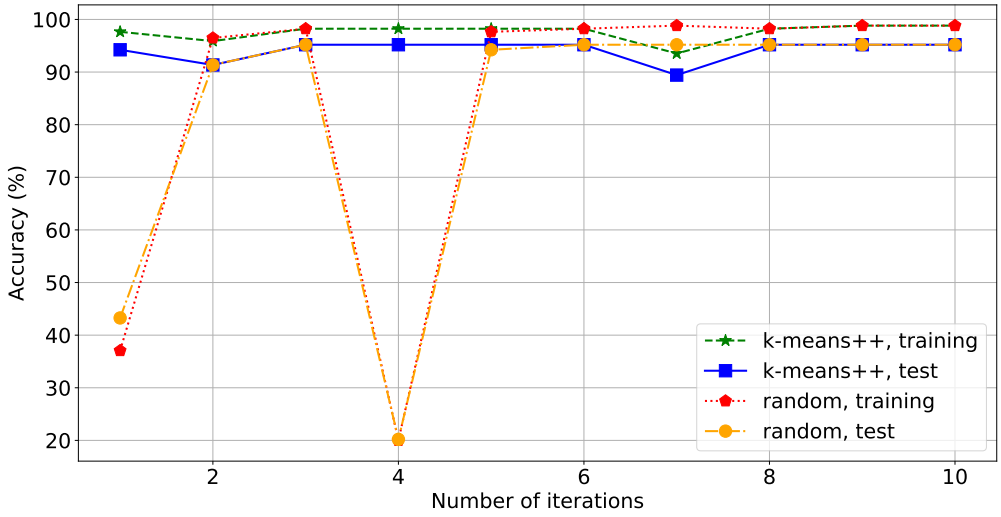
Phần này sẽ thực nghiệm với mô hình học máy *không giám sát* K-means Clustering. Với số cụm cố định là 3, chúng ta sẽ kiểm tra xem cách phân cụm của mô hình có gần với thực tế hay không. Một lần chạy, sẽ có 5 lượt khởi tạo tâm khác nhau, kết quả cuối cùng là cách phân cụm tốt nhất dựa trên hàm lỗi. Thuật toán sẽ dừng khi các tâm không di chuyển hay thuật toán đạt đến số lần lặp tối đa cho phép (lặp ở đây là một lần quét qua toàn bộ



Hình 25: Số lượng cá thể cánh cụt của từng cụm theo tỷ lệ.

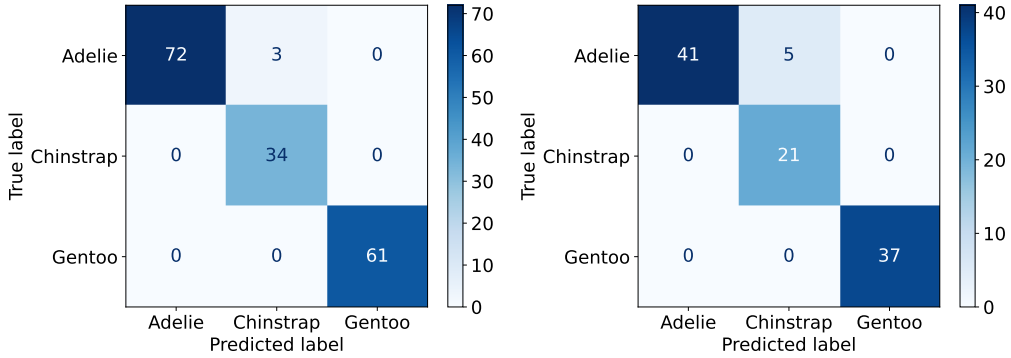
dữ liệu). Dữ liệu sử dụng đã chuẩn hóa, bộ huấn luyện có 170 mẫu.

Khi sử dụng k-means++ với 10 lần lặp trên bộ dữ liệu huấn luyện, ta có tỷ lệ theo số lượng mẫu trong mỗi cụm như trong Hình 25. Sự phân bố này khá gần với sự phân bố của số lượng mỗi loài chim cánh cụt trong bộ dữ liệu gốc (Hình 2). Do vậy, tiếp theo, chúng ta sẽ thử tiến hình đặt tên nhãn cho các cụm dựa theo thứ tự về số lượng của mỗi loài cánh cụt trong thực tế (cụm có nhiều mẫu nhất ứng với loài cánh cụt đông nhất trong bộ dữ liệu ban đầu, cụm có ít mẫu nhất ứng với loài chim cánh cụt ít nhất trong bộ dữ liệu ban đầu), rồi kiểm tra kết quả dự đoán đạt được trên cả bộ huấn luyện lẫn kiểm tra. Số lần lặp tối đa được thay đổi từ 1 đến 10, cách khởi tạo tâm cụng theo kiểu ngẫu nhiên và theo *k-means++* sẽ được so sánh với nhau. Có thể thấy trong Hình 26, số lần lặp ít có thể khiến mô hình dự đoán không tốt. Cách khởi tạo ngẫu nhiên có thể mang đến kết quả rất tồi, do đó sử dụng phương pháp cải tiến *k-means++* sẽ phù hợp hơn. Nói chung thì độ chính xác trên bộ huấn luyện cao hơn so với bộ kiểm tra, do các mẫu trên bộ huấn luyện được dùng trực tiếp để cập nhật tâm các cụm trong thuật toán.



Hình 26: Độ chính xác trên bộ dữ liệu huấn luyện và xác thực với số lần lặp khác nhau khi có hoặc không sử dụng k-means++.

Hình 27 biểu diễn ma trận lỗi (confusion matrix) khi dùng *k-means++* với 10 lần lặp trên bộ huấn luyện và bộ kiểm tra. Có thể thấy mô hình dễ dàng nhận biết được loài Gentoo. Khi quan sát Hình 3, các điểm tương ứng của loài chim cánh cụt Gentoo nằm khá cách biệt về mặt "vị trí" so với hai loài chim cánh cụt còn lại. Mô hình chỉ nhầm lẫn một số loài chim Adelie thành Chinstrap. Có thể giải thích hiện tượng này như sau, số lượng loài Adelie cao gấp hơn hai lần so với loài Chinstrap, do đó phạm vi phân bố của các giá trị từng đặc tính cũng rộng hơn, có thể thấy tương đối trong các biểu đồ phân tán hình 3.



(a) Training (b) Test

Hình 27: Ma trận lỗi khi dùng *k-means++* với 10 lần lặp trên bộ dữ liệu huấn luyện (trái) và bộ dữ liệu kiểm tra (phải).

Dù chỉ sử dụng kỹ thuật phân cụm đơn giản, với khoảng cách chỉ là Euclid, kết quả dự đoán đạt được đã rất cao (độ đo *precision*, *recall*, *f1* đều là 95.19 %). Dó đó dễ hiểu khi các mô hình học máy *có giám sát* đang mang lại kết quả gần như tuyệt đối.

5 Kết luận

Trong quá trình làm bài tập lớn, nhóm đã thực hiện được các công việc sau:

- Tìm hiểu về bài toán phân loại đa lớp.
- Tìm hiểu cách ứng dụng năm mô hình học máy giải quyết bài toán đặt ra.
- Tiến hành các thực nghiệm để đo độ hiệu quả của các cách mô hình đề xuất.
- Phân tích, bàn luận về các kết quả đạt được.

Các khó khăn gặp phải chủ yếu liên quan đến tiền xử lý dữ liệu và tìm hiểu cách áp dụng các mô hình trong framework **scikit-learn** và thư viện **Keras**.

Trong tương lai, nhóm muốn thực nghiệm các mô hình đã sử dụng trên các bài toán phân loại đa lớp khác phức tạp hơn, đồng thời cũng tìm hiểu theo các mô hình khác.

Để hoàn thành được báo cáo này, nhóm chúng em xin gửi lời cảm ơn sâu sắc đến PGS. TS. Thân Quang Khoát vì những tiết dạy tuyệt vời và đáng quý của thầy ở môn "Nhập môn Học máy và Khai phá dữ liệu".

Toàn bộ mã nguồn trong báo cáo được công khai tại link:

https://github.com/Tahuubinh/ML_HUST_Project

Tài liệu tham khảo

[1] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2015.

[2] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[3] Pulse Headlines. Climate change is directly affecting the penguin population. https://www.pulseheadlines.com/climate-change-is-directly-affecting-the-penguin-population/27828.

[4] Kaggle. Palmer archipelago (antarctica) penguin data. https://www.kaggle.com/datasets/parulpandey/palmer-archipelago-antarctica-penguin-data.

[5] Kristen B Gorman, Tony D Williams, and William R Fraser. Ecological sexual dimorphism and environmental variability within a community of antarctic penguins (genus pygoscelis). *PloS one*, 9(3):e90081, 2014.

[6] Gaurav L Agrawal and Hitesh Gupta. Optimization of c4. 5 decision tree algorithm for data mining application. *International Journal of Emerging Technology and Advanced Engineering*, 3(3):341–345, 2013.

[7] Badr Hssina, Abdelkarim Merbouha, Hanane Ezzikouri, and Mohammed Erritali. A comparative study of decision tree id3 and c4. 5. *International Journal of Advanced Computer Science and Applications*, 4(2):13–19, 2014.

[8] Gaussian naive bayes. https://iq.opengenus.org/gaussian-naive-bayes/.

[9] James Cameron Patterson. *Managing a real-time massively-parallel neural architecture*. The University of Manchester (United Kingdom), 2012.

[10] Activation functions in neural network. https://studymachinelearning.com/activation-functions-in-neural-network.

[11] Roza Dastres and Mohsen Soori. Artificial neural network systems. *International Journal of Imaging and Robotics (IJIR)*, 21(2):13–25, 2021.

[12] GateVidyalay. K-means clustering. https://www.gatevidyalay.com/tag/k-means-clustering-flowchart.

[13] Normalized Nerd. Random forest algorithm clearly explained! https://www.youtube.com/watch?v=v6VJ2RO66Ag.

[14] Jeremy Jordan. Setting the learning rate of your neural network. https://www.jeremyjordan.me/nn-learning-rate.