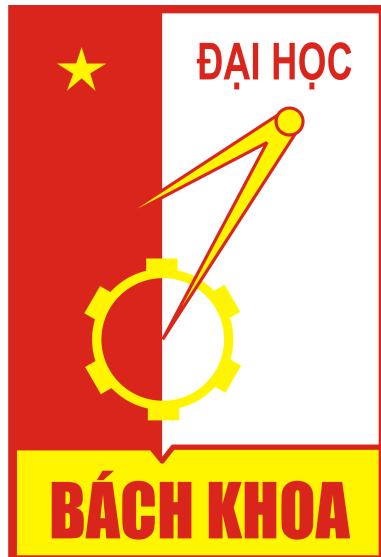


**Đại học Bách khoa Hà Nội**  
**Trường công nghệ Thông tin và Truyền thông**



**Bài tập lớn**

Môn: Nhập môn Học máy và Khai phá dữ liệu

*Tên đề tài:* Dự đoán loài các loài hoa

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

*Nhóm sinh viên thực hiện:*

Võ Minh Trí 20210862

Nguyễn Đức Thịnh 20210817

Vũ Trịnh Kim 20215409

*Mã lớp:* 141320

*Mã HP:* IT3190

Hà Nội, 22 tháng 6 năm 2023

# Mục lục

<b>Lời mở đầu</b>	<b>1</b>
<b>1 Giới thiệu chung</b>	<b>2</b>
<b>2 Dữ liệu và tiền xử lý dữ liệu</b>	<b>3</b>
2.1 Dữ liệu . . . . .	3
2.2 Tiền xử lý dữ liệu . . . . .	3
2.2.1 Ý tưởng của bài toán . . . . .	3
2.2.2 Nguyên lý của Convolutional Neural Networks . . . . .	5
2.2.3 Trình tự xử lý dữ liệu . . . . .	10
<b>3 Mô hình Học máy đề xuất</b>	<b>11</b>
3.1 SVM . . . . .	11
3.2 K-nearest Neighbors . . . . .	14
3.3 eXtreme Gradient Boosting(XGboost) . . . . .	15
3.3.1 Decision tree . . . . .	15
3.3.2 XGboost . . . . .	17
3.4 Ridge Classification . . . . .	19
<b>4 Kết quả thực nghiệm</b>	<b>21</b>
4.1 Mô hình huấn luyện trên bộ có màu và dự đoán tập ảnh có màu . . . . .	22
4.1.1 K-nearest Neighbor . . . . .	22
4.1.2 eXtreme Gradient Boosting (Xgboost) . . . . .	24
4.1.3 Support Vector Machine (SVM) . . . . .	26
4.1.4 Ridge classification . . . . .	28
4.1.5 So sánh kết quả các thuật toán có giám sát trên bộ dữ liệu kiểm tra .	29
4.2 Các mô hình được huấn luyện và phán đoán trên tập dữ liệu không màu . .	32
4.3 Mô hình học trên tập dữ liệu không màu và phán đoán trên tập có màu . .	35
4.3.1 KNN . . . . .	35
4.3.2 SVM . . . . .	35
4.3.3 XGboost . . . . .	36
4.3.4 Ridge classification . . . . .	36
4.3.5 Tổng quan kết quả phán đoán của các mô hình . . . . .	37
<b>5 Kết luận</b>	<b>38</b>



# 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). Trong những năm gần đây, lĩnh vực này đã có sự phát triển đáng kể. Không chỉ ở các quốc gia có nền kinh tế phát triển hàng đầu trên thế giới, mà ngay cả ở các quốc gia đang phát triển như Việt Nam, ứng dụng của học máy cũng đang ngày càng gia tăng, góp phần tạo ra những tác động tích cực đáng kể trong 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 "**Phân loại các loài hoa**". Để giải quyết bài toán này và đánh giá hiệu quả, nhóm đã áp dụng một số mô hình học máy, bao gồm cả mô hình có *giám sát* và mô hình *không giám sát*. Các mô hình này sẽ được thử nghiệm và đánh giá thông qua các thực nghiệm thực tế.

Báo cáo bao gồm 5 phần chính như sau:

- **Phần 1:** Giới thiệu về Machine Learning và Data Mining nói chung và bài toán "Phân loại các loài hoa"
- **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 toán
- **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 ?. 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 rõ-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 ?. 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 ??) 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.

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 đặc 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 đặc 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 sát như K-nearest Neighbor, SVM, XGboost
- Tiến hành các thực nghiệm để đánh giá các mô hình. Tham số chính là độ chính xác của các dự 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 các loài hoa từ trang web Kaggle ? kết hợp với những dữ liệu cào từ trên mạng để cân đối tập dữ liệu. Tập dữ liệu chứa thông tin về 8 loài hoa khác nhau: hoa chuông, hoa cúc, hoa bồ công anh, hoa tulip, hoa hướng dương, hoa hồng, hoa sen, hoa diên vĩ; với khoảng 20000 ảnh được phân bố đồng đều với 2500 ảnh mỗi loại.

Trong bài toán có hoa bồ công anh với hai hình thái phân biệt: *hình thái non trưởng thành* và *hình thái trưởng thành* trong hình 1. Mục tiêu là đánh giá khả năng dự đoán của mô hình với các hình thái này.



Hình 1: Các giai đoạn phát triển của hoa bồ công anh

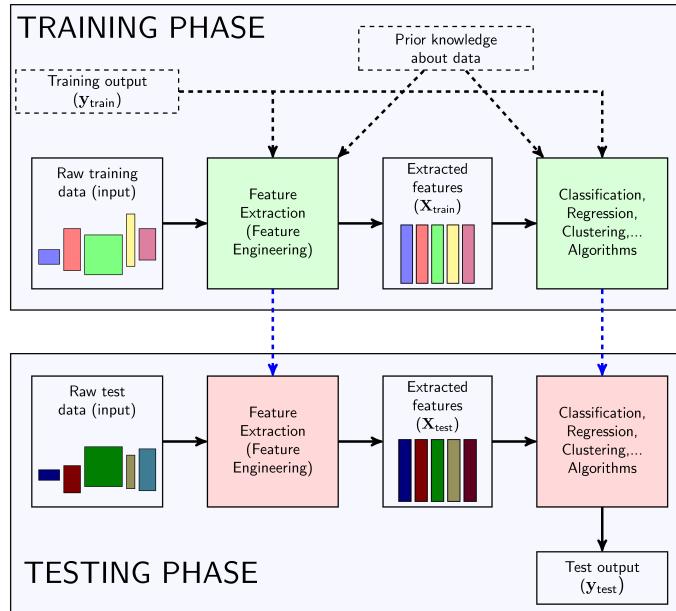
### 2.2 Tiền xử lý dữ liệu

#### 2.2.1 Ý tưởng của bài toán

Với các bài toán về Computer Vision, các bức ảnh là các ma trận có kích thước khác nhau. Chúng ta cần phải tìm một phép biến đổi để loại ra những dữ liệu nhiễu (noise), và để đưa dữ liệu thô với số chiều khác nhau về cùng một chuẩn (cùng là các vector hoặc ma trận). Dữ liệu chuẩn mới này phải đảm bảo giữ được những thông tin đặc trưng (features) cho dữ liệu thô ban đầu. Không những thế, tùy vào từng bài toán, ta cần thiết kế những

phép biến đổi để có những features phù hợp. Quá trình quan trọng này được gọi là Feature Extraction, hoặc Feature Engineering.

“ Coming up with features is difficult, time-consuming, requires expert knowledge. “Applied machine learning” is basically feature engineering. ” (Andrew Ng)



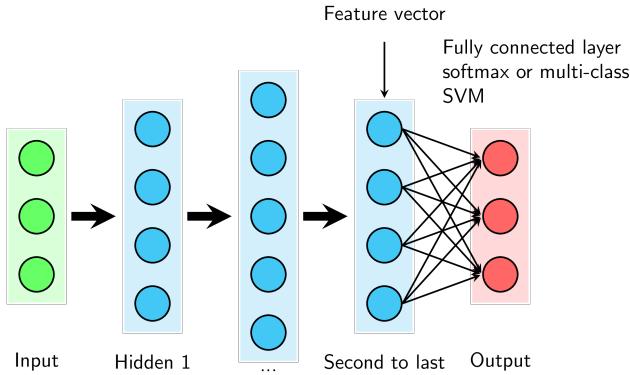
Hình 2: Mô hình chung cho các bài toán Machine Learning.

Trước Deep Learning, bài toán phân loại ảnh thường được chia thành 2 bước: Feature Engineering và Train a Classifier. Hai bước này thường được tách rời nhau. Với Feature Engineering, các phương pháp thường được sử dụng cho ảnh là SIFT (Scale Invariant Feature Transform), SURF (Speeded-Up Robust Features), HOG (Histogram of Oriented Gradients), LBP (Local Binary Pattern), .... Các Classifier thường được sử dụng là multi-class SVM, Softmax Regression, Random Forest, ....

Các phương pháp Feature Engineering nêu trên thường được gọi là các hand-crafted features (feature được tạo thủ công) vì nó chủ yếu dựa trên các quan sát về đặc tính riêng của ảnh. Các phương pháp này vẫn còn nhiều hạn chế vì quá trình tìm ra các features và các classifier phù hợp vẫn là riêng biệt.

Những năm gần đây, các mạng học sâu phát triển cực nhanh dựa trên lượng dữ liệu training khổng lồ và khả năng tính toán ngày càng được cải tiến của các máy tính. Các kết quả cho bài toán phân loại ảnh ngày càng được nâng cao. Nhìn chung, các mô hình học sâu này đều bao gồm rất nhiều layers. Các layers phía trước thường là các Convolutional layers kết hợp với các nonlinear activation functions và pooling layers (và được gọi chung là mạng tích chập). Layer cuối cùng là một Fully Connected Layer và thường là một Softmax

Regression có số lượng units bằng với số lượng classes như hình 3. Vì vậy output ở layer gần cuối cùng (second to last layer) có thể được coi là feature vectors và Softmax Regression chính là Classifier được sử dụng. Sự hiệu quả của mạng học sâu là cả bộ trích chọn đặc trưng (feature extractor) và bộ phân lớp (classifier) được huấn luyện đồng thời và hỗ trợ lẫn nhau.



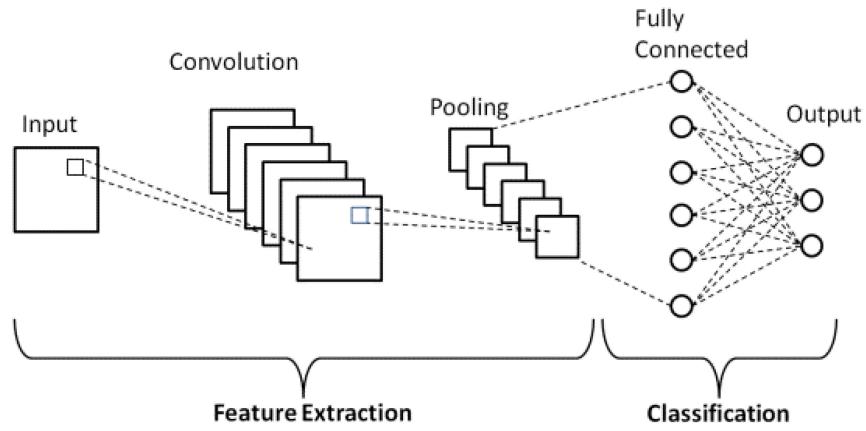
Hình 3: Mô hình chung cho các bài toán classification sử dụng Deep Learning

Chính nhờ việc features và classifier được trained cùng nhau qua deep networks khiến cho các mô hình này đạt kết quả tốt. Tuy nhiên, những mô hình này đều là các Deep Networks với rất nhiều layers. Việc training dựa trên 1.2M bức ảnh của ImageNet cũng tốn rất nhiều thời gian (2-3 tuần). Với các bài toán dựa trên tập dữ liệu khác, rất ít khi người ta xây dựng và train lại toàn bộ Network từ đầu, bởi vì có rất ít các cơ sở dữ liệu có kích thước lớn. Thay vào đó, chúng ta có thể sử dụng các mô hình (nêu phía trên) đã được trained từ trước, và sử dụng một vài mô hình học máy khác để giải quyết bài toán.

Như đã đề cập, toàn bộ các layer trừ output layer có thể được coi là một bộ Feature Extractor. Dựa trên nhận xét rằng các bức ảnh đều có những đặc tính giống nhau nào đó, với cơ sở dữ liệu khác, ta cũng có thể sử dụng phần Feature Extractor này để tạo ra các feature vectors. Sau đó, ta thay output layer bằng một vài mô hình phân loại khác. Ta chỉ cần train các mô hình này. Việc làm này đã tăng kết quả phân lớp lên rất nhiều so với việc sử dụng các hand-crafted features.

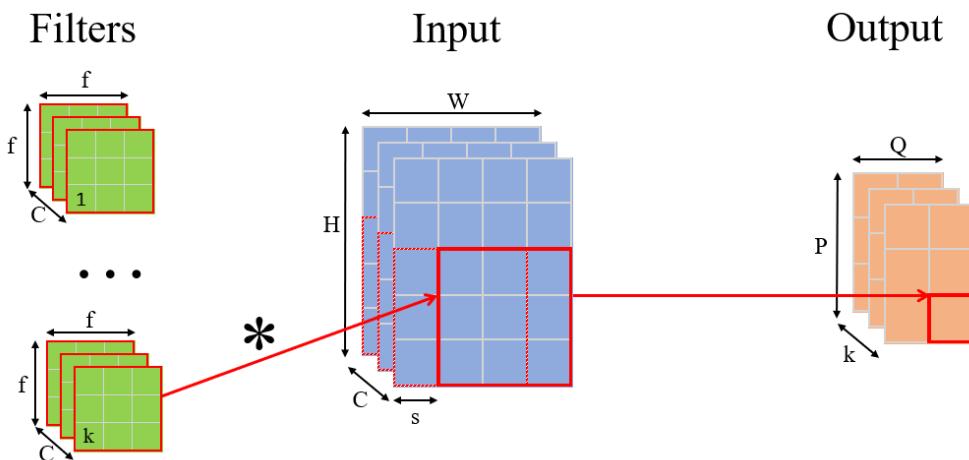
### 2.2.2 Nguyên lý của Convolutional Neural Networks

Mạng nơ-ron tích chập (CNN) là một kiến trúc bao gồm nhiều lớp tích chập. Trong CNN, mỗi lớp tích chập liên tiếp tạo ra một trùu tượng cấp cao hơn của dữ liệu đầu vào, được gọi là bản đồ đặc trưng, lưu trữ những thông tin thiết yếu nhưng duy nhất. CNN hiện đại có thể đạt được hiệu suất vượt trội trong nhiều tác vụ bằng cách sử dụng kiến trúc phân tầng lớp rất sâu. Trong các tác vụ thị giác máy tính, các kiến trúc CNN hiện đại thường có từ 5 đến hơn 1000 lớp tích chập.



Hình 4: Minh họa kiến trúc đơn giản của CNN

Các lớp phổ biến **Lớp tích chập (Convolution layer)** Lớp tích chập sử dụng các bộ lọc để thực hiện phép tích chập khi đưa chúng đi qua đầu vào theo các chiều của nó.



Hình 5: Minh họa tín hiệu vào và ra của lớp tích chập

### Bộ lọc (Filter)

Một bộ lọc kích thước  $f \times f$  áp dụng lên đầu vào với  $C$  kênh (channels) thì có kích thước tổng là  $f \times f \times C$  và thực hiện phép tích chập trên đầu vào kích thước  $H \times W \times C$ , cho ra kết quả là một bản đồ đặc trưng (feature map) có kích thước  $P \times Q \times 1$ .

Với  $k$  bộ lọc kích thước  $f \times f$ , kết quả là một bản đồ đặc trưng với kích thước  $P \times Q \times k$ .

### Bước nhảy (Stride)

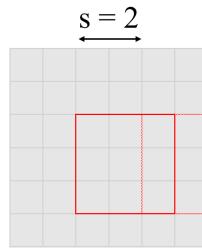
Đối với phép tích chập hoặc phép tổng hợp, bước nhảy  $s$  ký hiệu số pixel mà cửa sổ sẽ di chuyển sau mỗi lần thực hiện phép tính.

Trong đó :  $s > 1$ , hình ảnh kết quả nhỏ hơn hình ảnh đầu vào.

$f \geq s$ , hình ảnh được bao phủ toàn bộ.

### **Đệm (Padding)**

Một vấn đề khi áp dụng các tầng tích chập là việc có thể mất số điểm ảnh trên biên của ảnh. Nếu áp dụng nhiều phép tích chập liên tiếp, mọi thông tin có ích trên viền của ảnh gốc sẽ bị xóa sạch. Đệm là công cụ phổ biến để xử lý vấn đề này.



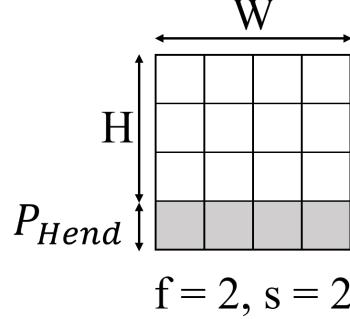
Hình 6: Minh họa việc mất điểm ảnh với  $s = 2, f = 3$ , kích thước ảnh:  $6 \times 6$

Đệm là chèn thêm các điểm ảnh xung quanh đường biên trên bức ảnh đầu vào, nhờ đó làm tăng kích thước sử dụng của ảnh. Thông thường, các giá trị điểm ảnh thêm vào là 0.

- Valid: không sử dụng padding, bỏ phép tích chập cuối nếu số chiều không khớp.

$$P = 0$$

- Same: sử dụng padding để làm cho bản đồ đặc trưng có kích thước  $\lceil \frac{H}{s} \rceil \times \lceil \frac{W}{s} \rceil$



Hình 7: Minh họa phương pháp Same padding

$$P_{Hstart} = \lfloor \frac{s[\frac{H}{s}] - H + f - s}{2} \rfloor \quad P_{Hend} = \lceil \frac{s[\frac{H}{s}] - H + f - s}{2} \rceil$$

$$P_{Wstart} = \lfloor \frac{s[\frac{W}{s}] - W + f - s}{2} \rfloor \quad P_{Wend} = \lceil \frac{s[\frac{W}{s}] - W + f - s}{2} \rceil$$

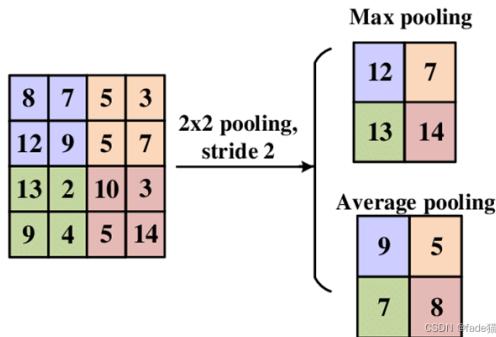
**Mối quan hệ của các tham số trong tầng tích chập**

$$P = \frac{H-f+P_{Hstart}+P_{Hend}}{s} + 1 \quad Q = \frac{W-f+P_{Wstart}+P_{Wend}}{s} + 1$$

## Lớp tổng hợp (Pooling layer)

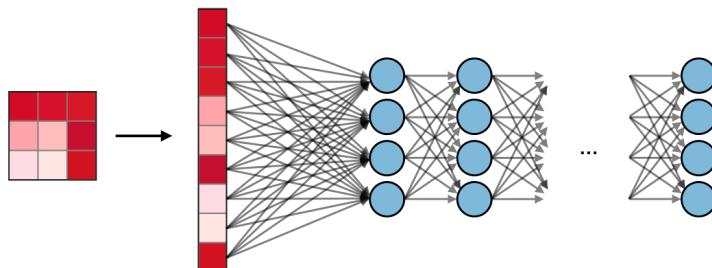
Lớp tổng hợp thường được sử dụng sau lớp tích chập nhằm giản hóa thông tin đầu ra, giảm bớt kích thước bản đồ đặc trưng. Phép tổng hợp được áp dụng lên từng kênh. Với đầu vào có kích thước  $H \times W \times C$ , lớp tổng hợp cho ra kết quả là một bản đồ đặc trưng có kích thước  $P \times Q \times C$ . Có hai loại pooling phổ biến:

- Max pooling: từng phép pooling chọn giá trị lớn nhất trong khu vực mà nó được áp dụng, điều này bảo toàn được các đặc trưng đã phát hiện.
- Average pooling: từng phép pooling tính trung bình các giá trị trong khu vực mà nó đang được áp dụng.



Hình 8: Minh họa Max pooling và Average pooling

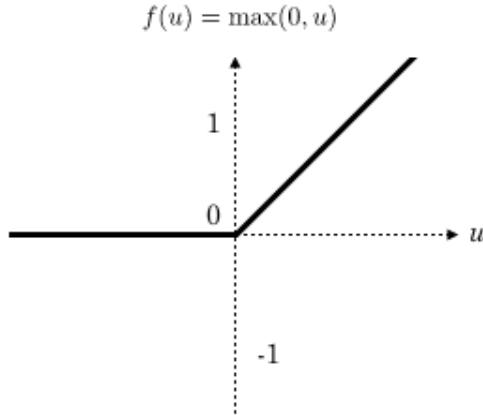
**Lớp kết nối đầy đủ (Fully connected layer)** Lớp kết nối đầy đủ nhận đầu vào là các dữ liệu đã được làm phẳng, mà mỗi đầu vào đó được kết nối đến tất cả neuron. Trong CNNs, các lớp kết nối đầy đủ thường được sử dụng ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng ví dụ như độ chính xác của lớp.



Hình 9: Minh họa lớp kết nối đầy đủ

Các hàm kích hoạt phổ biến

**ReLU (Rectified Linear Unit)** là một hàm kích hoạt  $f$  được sử dụng trên tất cả thành phần nhằm tăng tính phi tuyến của mạng



Hình 10: Minh họa ReLU

*Việc tạo sự phi tuyến cho mô hình là cần thiết, ví dụ với tập dữ liệu có phân bố phức tạp thì việc sử dụng một hàm tuyến tính là không đủ để biểu diễn*

**Softmax** có thể được coi là một hàm logistic tổng quát lấy đầu vào là một vector chứa các giá trị  $x \in \mathbb{R}^n$  và cho ra một vector gồm các xác suất  $p \in \mathbb{R}^n$ .

$$\text{Trong đó: } p = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} \text{ với } p_i = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$$

Hàm Softmax giới hạn giá trị đầu ra trong khoảng  $(0, 1)$ , điều này giúp kết quả của các phép toán không bị rất lớn hoặc rất bé gây ra những vấn đề về mặt tính toán và mạng có thể khó hội tụ.

### Các kỹ thuật phổ biến Chuẩn hóa batch

Kiểm soát phân phối đầu vào giữa các lớp mạng có thể giúp tăng tốc đáng kể việc huấn luyện và cải thiện được hiệu suất chung. Theo đó, phân phối của đầu vào của lớp  $(\sigma, \mu)$  được chuẩn hóa sao cho nó có giá trị trung bình bằng 0 và độ lệch chuẩn đơn vị.

$$\text{Trong đó: } y = BN(x) = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

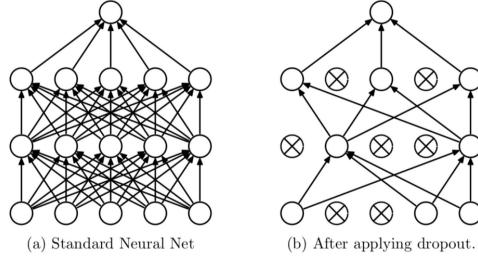
Trong đó:  $\epsilon$  là một hằng số nhỏ nhằm tránh các vấn đề về số

$\gamma, \beta$  là các tham số được mạng học trong quá trình huấn luyện với  $\gamma$  giúp điều chỉnh phương sai phân phối và  $\beta$  là độ lệch (bias) giúp dịch chuyển phân phối sang trái hay phải.

### Dropout

Drop out là một kỹ thuật để khắc phục hiện tượng overfitting. Trong quá trình huấn

luyện, một số đặc trưng đầu ra có thể bị bỏ qua thông qua lớp dropout một cách ngẫu nhiên với tỉ lệ bỏ ngẫu nhiên là  $p$ . Điều này làm mạng giảm việc bị phụ thuộc vào các đặc trưng có giá trị lớn hơn nhiều so với các đặc trưng còn lại, vốn sẽ được coi là thông tin quan trọng và gán trọng số lớn, trong khi đó các thông tin khác có thể sẽ bị coi là nhiễu và không được học, dẫn đến mạng học bằng cách ghi nhớ các ví dụ và kém mang tính tổng quát hơn.



Hình 11: Minh họa Dropout

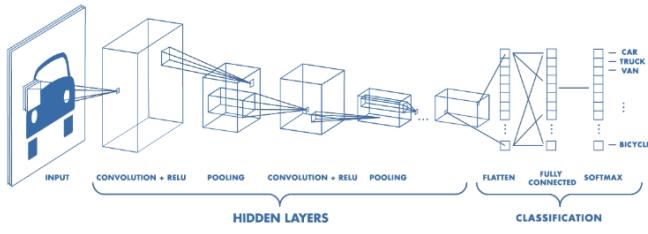
### 2.2.3 Trình tự xử lý dữ liệu

Vì dữ liệu thu được từ những nguồn khác nhau với kích thước không đồng nhất nên bước đầu tiên cần làm là đưa toàn bộ tập dữ liệu về cùng một kích cỡ. Trong bài báo cáo, dữ liệu sẽ chia làm hai cách để xử lý chính: để nguyên màu và lọc bỏ màu trong ảnh bằng bộ lọc lapcian như hình 12



Hình 12: Sau khi loại bỏ màu

Hình ảnh sau đó sẽ được chuyển sang dạng mảng có kích thước  $224*224*3$  rồi được đưa vào một nơ-ron tích chập (CNN) để giảm chiều và trích xuất các thuộc tính của ảnh, dữ liệu sau khi qua xử lý sẽ có dạng vector 1000 chiều. Mô hình mạng CNN có cấu trúc như hình 13, dữ liệu sử dụng trong bài lấy output ở lớp flatten.



Hình 13: Mô hình mạng cnn sử dụng

### 3 Mô hình Học máy để xuất

Phần này của báo cáo sẽ trình bày nội dung lý thuyết của bốn mô hình học máy có giám sát (SVM, K-nearest Neighbor, XGBoost và Ridge Classification)

#### 3.1 SVM

Support vector machine (SVM) là một trong những thuật toán phân lớp phổ biến và hiệu quả. Trong thuật toán này, chúng ta vẽ đồ thị dữ liệu là các điểm trong n chiều (ở đây n là số lượng các tính năng bạn có) với giá trị của mỗi tính năng sẽ là một phần liên kết. Sau đó chúng ta thực hiện tìm "siêu phẳng" (hyper-plane) phân chia các lớp. Hyper-plane nó chỉ hiểu đơn giản là 1 đường thẳng có thể phân chia các lớp ra thành hai phần riêng biệt. Ý tưởng đứng sau SVM có vẻ khá đơn giản, nhưng để hiểu được cách tìm nghiệm của nó, chúng ta cần nhớ lại một chút kiến thức về tối ưu và duality.

- Khoảng cách từ một điểm tới một siêu phẳng: Trong không gian 2 chiều, ta biết rằng khoảng cách từ một điểm có tọa độ  $(x_0, y_0)$  tới đường thẳng có phương trình  $w_1x + w_2y + b = 0$  được xác định bởi:

$$\frac{|w_1x_0 + w_2y_0 + b|}{\sqrt{w_1^2 + w_2^2}}$$

Trong không gian ba chiều, khoảng cách từ một điểm có tọa độ  $(x_0, y_0, z_0)$  tới một mặt phẳng có phương trình  $w_1x + w_2y + w_3z + b = 0$  được xác định bởi:

$$\frac{|w_1x_0 + w_2y_0 + w_3z_0 + b|}{\sqrt{w_1^2 + w_2^2 + w_3^2}}$$

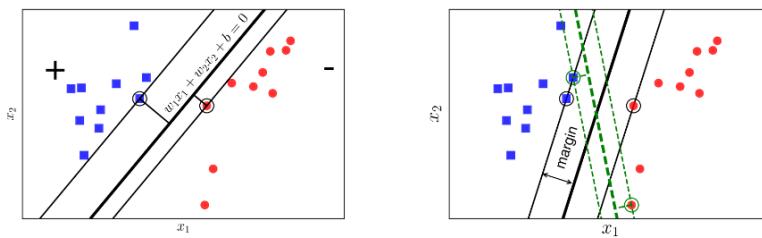
Việc này có thể được tổng quát lên không gian nhiều chiều: Khoảng cách từ một điểm (vector) có tọa độ  $\mathbf{x}_0$  tới siêu phẳng (hyperplane) có phương trình  $\mathbf{w}^T \mathbf{x} + b = 0$  được

xác định bởi:

$$\frac{|\mathbf{w}^T \mathbf{x}_0 + b|}{\|\mathbf{w}\|_2}$$

Với  $\|\mathbf{w}\|_2 = \sqrt{\sum_{i=1}^d w_i^2}$  với  $d$  là số chiều của không gian.

- Chúng ta cùng quay lại với bài toán trong Perceptron Learning Algorithm (PLA). Giả sử có hai class khác nhau được mô tả bởi các điểm trong không gian nhiều chiều, hai classes này linearly separable, tức là tồn tại một siêu phẳng phân chia chính xác hai classes đó. Cần tìm một siêu phẳng phân chia hai classes đó. Chúng ta đã biết rằng, thuật toán PLA có thể làm được việc này nhưng nó có thể cho chúng ta vô số nghiệm. Câu hỏi đặt ra là: trong vô số các mặt phân chia đó, đâu là mặt phân chia tốt nhất theo một tiêu chuẩn nào đó?



Hình 14: Margin của hai classes là bằng nhau và lớn nhất có thể.

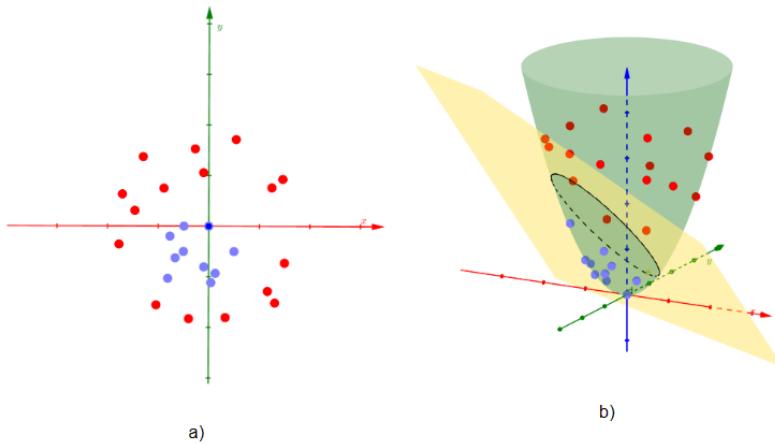
Ở Hình 2 trái, đường phân chia gần class tròn đỏ hơn class vuông xanh rất nhiều. Chúng ta cần một đường phân chia sao cho khoảng cách từ điểm gần nhất của mỗi class (các điểm được khoanh tròn) tới đường phân chia là như nhau. Khoảng cách như nhau này được gọi là margin (lề). Chúng ta xét tiếp Hình 2 bên phải khi khoảng cách từ đường phân chia tới các điểm gần nhất của mỗi class là như nhau. Xét hai cách phân chia bởi đường nét liền màu đen và đường nét đứt màu lục, ta thấy rõ ràng đường nét liền màu đen sẽ phân lớp tốt hơn vì nó tạo ra một margin rộng hơn.

Việc margin rộng hơn sẽ mang lại hiệu ứng phân lớp tốt hơn vì sự phân chia giữa hai classes là rạch ròi hơn. Việc này là một điểm khá quan trọng giúp SVM mang lại kết quả phân loại tốt hơn so với PCA. Bài toán tối ưu trong SVM chính là bài toán đi tìm đường phân chia sao cho margin là lớn nhất. Đây cũng là lý do vì sao SVM còn được gọi là Maximum Margin Classifier.

Với những bài toán mà dữ liệu giữa hai classes là hoàn toàn không linear separable (không phân biệt tuyến tính), chúng ta sử dụng Kernel SVM. Ý tưởng cơ bản của Kernel SVM và các phương pháp kernel nói chung là tìm một phép biến đổi sao cho dữ liệu ban đầu là không

phân biệt tuyến tính được biến sang không gian mới. Ở không gian mới này, dữ liệu trở nên phân biệt tuyến tính.

Xét ví dụ dưới đây với việc biến dữ liệu không phân biệt tuyến tính trong không gian hai chiều thành phân biệt tuyến tính trong không gian ba chiều bằng cách giới thiệu thêm một chiều mới:



Hình 15: Ví dụ về Kernel SVM

Nói một cách ngắn gọn, Kernel SVM là việc đi tìm một hàm số biến đổi dữ liệu  $x$  từ không gian feature ban đầu thành dữ liệu trong một không gian mới bằng hàm số  $\phi()$ . Trong ví dụ này, hàm  $\phi()$  đơn giản là giới thiệu thêm một chiều dữ liệu mới (một feature mới) là một hàm số của các features đã biết. Hàm số này cần thỏa mãn mục đích của chúng ta: trong không gian mới, dữ liệu giữa hai classes là phân biệt tuyến tính hoặc gần như phân biệt tuyến tính. Nếu phải so sánh, ta có thể thấy rằng hàm biến đổi  $\phi()$  tương tự như activation functions trong Neural Networks. Tuy nhiên, có một điểm khác biệt ở đây là: trong khi nhiệm vụ của activation function là phá vỡ tính tuyến tính của mô hình, hàm biến đổi  $\phi()$  đi biến dữ liệu không phân biệt tuyến tính thành phân biệt tuyến tính.

Các hàm  $\phi()$  thường tạo ra dữ liệu mới có số chiều cao hơn số chiều của dữ liệu ban đầu, thậm chí là vô hạn chiều. Nếu tính toán các hàm này trực tiếp, chắc chắn chúng ta sẽ gặp các vấn đề về bộ nhớ và hiệu năng tính toán. Có một cách tiếp cận là sử dụng các kernel functions mô tả quan hệ giữa hai điểm dữ liệu bất kỳ trong không gian mới, thay vì đi tính toán trực tiếp từng điểm dữ liệu trong không gian mới.

Radial Basic Function (RBF) kernel hay Gaussian kernel được sử dụng nhiều nhất trong thực tế, và là lựa chọn mặc định của SVM. Nó thể hiện tốt khi chúng ta chưa biết

phân bố của dữ liệu, ví dụ như data dạng ảnh. Nó được định nghĩa bởi:

$$k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma \|\mathbf{x} - \mathbf{z}\|_2^2), \quad \gamma > 0$$

Khi sử dụng kernel này, mô hình của chúng ta có hai trọng số chính là C và gamma.

### 3.2 K-nearest Neighbors

Thuật toán K-nearest neighbors (KNN) là một thuật toán supervised-learning đơn giản và hiệu quả trong một số trường hợp trong Machine Learning. KNN không học gì từ dữ liệu huấn luyện trong quá trình huấn luyện, mà tính toán chỉ được thực hiện khi cần dự đoán kết quả cho dữ liệu mới. Do đó, KNN được xếp vào loại thuật toán lazy learning. KNN có thể được áp dụng cho cả bài toán Classification và Regression trong Supervised learning. Nó còn được gọi là thuật toán Instance-based hay Memory-based learning.

Trong bài toán Classification, KNN suy ra nhãn (label) của một điểm dữ liệu mới dựa trên k điểm dữ liệu gần nhất trong tập huấn luyện. Nhãn của dữ liệu kiểm thử có thể được quyết định thông qua *major voting* (bầu chọn theo số phiếu) của các điểm gần nhất, hoặc có thể được suy ra bằng cách đánh trọng số khác nhau cho các điểm gần nhất rồi suy ra nhãn.

Có hai thành phần chính trong KNN: lựa chọn số hàng xóm (k) và cách tính khoảng cách giữa hai điểm dữ liệu. Thông thường, chúng ta nên sử dụng k lớn hơn để dự đoán ( $k > 1$ ), nhưng không nên chọn quá nhiều. Nếu chọn quá nhiều hàng xóm, có thể làm mất cấu trúc ban đầu của dữ liệu và dẫn đến dự đoán không tốt.

Trong không gian một chiều, khoảng cách giữa hai điểm được tính bằng giá trị tuyệt đối của 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 khác nhau. Trong KNN, phép đo khoảng cách đóng vai trò quan trọng. Có nhiều công thức thường được sử dụng để đo khoảng cách, ví dụ như: Minkowski ( $L_p$  – norm), Manhattan ( $L_1$  – norm), Euclid ( $L_2$  – norm), ...

Trong KNN, việc chuẩn hóa các thuộc tính đôi khi rất quan trọng để có được khả năng dự đoán tốt. Nếu dữ liệu không được chuẩn hóa, 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.

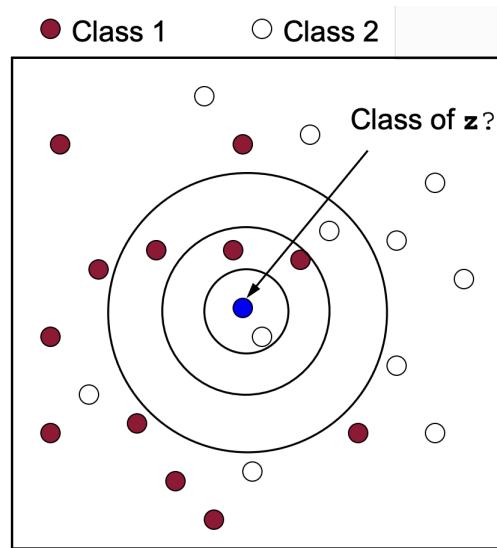
#### KNN cho bài toán Classification

- 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  $x_i = (x_{i1}, x_{i2}, \dots, x_{in})^T$

- 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 16 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 16: Ví dụ về bài toán KNN.

### 3.3 eXtreme Gradient Boosting(XGboost)

Thuật toán XGboost 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.

#### 3.3.1 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 17: 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  $x_i = (x_{i1}, x_{i2}, \dots, x_{in})^T$ ,  $x_{i1} \in \{high, normal\}$ ,  $x_{i2} \in \{male, female, other\}$
- Có tập C là tập các nhãn đã được gán trước

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

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

- 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

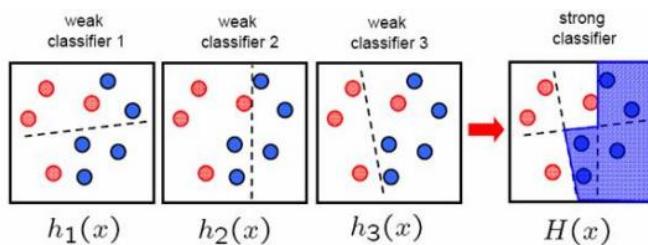
- Cho dữ liệu cần phán đoán duyệt qua cây này, dùng các thuộc tính của nó duyệt từ gốc đến lá, lấy nhãn ở lá để phán đoán cho dữ liệu mới

### 3.3.2 XGboost

#### Định nghĩa

XGBoost (Extreme Gradient Boosting) được phát triển bởi Tianqi Chen vào năm 2014, trong cuộc thi "Kaggle Higgs Machine Learning Challenge". XGBoost đã nhanh chóng trở thành một công cụ quan trọng trong cộng đồng học máy và phân tích dữ liệu. Nó đã giành được nhiều giải thưởng trong các cuộc thi học máy và trở thành một trong những thuật toán phổ biến nhất cho các bài toán phân loại và dự đoán trên các nền tảng như Kaggle.

Ý tưởng: XGBoost là một phương pháp tăng cường (boosting) dựa trên gradient, tức là nó tập trung vào việc tối ưu hóa hàm mất mát bằng cách thực hiện việc huấn luyện nhiều cây quyết định (weak learner) theo từng bước. Kết quả cuối cùng được lấy bằng tổng có trọng số của tất cả các dự đoán riêng lẻ như hình 18



Hình 18: Minh họa mô hình XGboost

Phát biểu bài toán :

- $y$  là biến ngẫu nhiên “output” hay “response”.
- $F^*(\mathbf{x})$  là hàm mục tiêu ánh xạ  $\mathbf{x}$  sang  $y$ .
- $(y_i, x_i)$  là mẫu dữ liệu “training”.
- $L(y, F(\mathbf{x}))$  là loss function:

Mục tiêu của thuật toán là tìm được hàm mục tiêu  $F^*$  sao cho cực tiểu hóa kỳ vọng của hàm lỗi.

$$F^* = \operatorname{argmin}_F E_{y, \mathbf{x}} L(y, F(\mathbf{x})) = \operatorname{argmin}_F E_{\mathbf{x}} [E_y (L(y, F(\mathbf{x}))) | \mathbf{x}]$$

Trong không gian tham số,  $F(\mathbf{x})$  thuộc lớp hàm  $F(\mathbf{x}; \mathbf{P})$ ,  $\mathbf{P} = P_1, P_2, \dots$  là tập hữu hạn các tham số khi đó dạng “additive” của lớp hàm này sẽ là:

$$F(\mathbf{x}; (\beta_m, \mathbf{a}_m)_1^M) = \sum_{m=1}^M \beta_m h(\mathbf{x}; \mathbf{a}_m)$$

Ta có thể hình dung  $(\mathbf{x}; \mathbf{a}_m)$  là một regression tree thứ m trong số M tree build được

từ dạng “addictive” này. Cụ thể, trong thuật toán học CART (Breiman, Friedman, Olshen, Stone 1983),  $\mathbf{a}_m$  là splitting variables, split locations, terminal node của từng cây.  $\beta_m$  chính là trọng số tương ứng đối với mỗi cây.

Để chọn được tham số cho mô hình  $F(\mathbf{x}; \mathbf{P})$  ta đi tính  $P^*$  sao cho:

$$\mathbf{P}^* = \operatorname{argmin}_{\mathbf{P}} \Phi(\mathbf{P})$$

trong đó:  $\Phi(\mathbf{P}) = E_{y, \mathbf{x}} L(y, F(\mathbf{x}; \mathbf{P}))$

khi đó:  $F^*(\mathbf{x}) = F(\mathbf{x}; \mathbf{P}^*)$

Để giải  $\mathbf{P}^*$  người ta thường biểu diễn tham số này dưới dạng sau:  $\mathbf{P}^* = \sum_{m=0}^M \mathbf{p}_m$

trong đó,  $\mathbf{p}_0$  là tham số khởi tạo và  $(\mathbf{p}_m)_1^M$  là successive increments (“steps” hay “boosts”). Nghĩa là, mỗi tham số hiện tại sẽ được tính dựa vào kết quả của chuỗi tham số tính được trước đó. Ta có thể thấy mô hình này tương tự như Random Forest điểm khác biệt duy nhất nằm ở chỗ training model như thế nào. Cũng như tất cả các thuật toán classification, ta sẽ định nghĩa hàm loss và đi optimize hàm này.

Lúc này, ta có thể sử dụng Steepest-descent để cực tiểu hoá bài toán tìm tham số này.

Đầu tiên, ta sẽ tính gradient  $\mathbf{g}_m$  như sau:

$$\mathbf{g}_m = \{g_{jm}\} = \left\{ \left[ \frac{\partial \Phi(\mathbf{P})}{\partial P_j} \right]_{\mathbf{P}=\mathbf{P}_{m-1}} \right\}$$

trong đó:  $\mathbf{P}_{m-1} = \sum_{i=0}^{m-1} \mathbf{p}_i$ .

Mỗi step được tính theo công thức:  $\mathbf{p}_m = -\rho_m \mathbf{g}_m$

trong đó:  $\rho_m = \operatorname{argmin}_{\rho} \Phi(\mathbf{P}_{m-1} - \rho \mathbf{g}_m)$ .

Dấu trừ trước gradient  $-\mathbf{g}_m$  cho biết đây là hướng “steepest-descent” mà thuật toán đang hướng tới để tìm điểm cực tiểu cục bộ.

Ở đây, ta áp dụng hướng tiếp cận “non-parametric” với phép tối thiểu hoá Steepest-descent cho không gian hàm số thay vì trong không gian tham số. Ta xem  $F(\mathbf{x})$  là “parameter” cần tìm để cực tiểu hoá:

$$\Phi(F) = E_{y, \mathbf{x}} L(y, F(\mathbf{x})) = E_{\mathbf{x}} [E_y (L(y, F(\mathbf{x}))) | \mathbf{x}],$$

tương đương với:

$$\phi(F(\mathbf{x})) = E_y [L(y, F(\mathbf{x})) | \mathbf{x}]$$

Với tập dữ liệu hữu hạn  $F(\mathbf{x}_i)_1^N$  hàm cần tìm sẽ là:

$$F^*(\mathbf{x}) = \sum_{m=0}^M f_m(\mathbf{x})$$

trong đó,  $f_0(\mathbf{x})$  là hàm khởi tạo, và  $\{f_m(\mathbf{x})\}_1^M$  là incremental functions (“steps” hay “boosts”) được định nghĩa tại mỗi bước steepest-descent như sau:

$$f_m(\mathbf{x}) = -\rho_m g_m(\mathbf{x})$$

$$\text{với: } g_m(\mathbf{x}) = \left[ \frac{\partial \Phi(F(\mathbf{x}))}{\partial F(\mathbf{x})} \right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})} = \left[ \frac{\partial E_y[L(y, F(\mathbf{x})) | \mathbf{x}]}{\partial F(\mathbf{x})} \right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}$$

$$\text{và } F_{m-1}(\mathbf{x}) = \sum_{i=0}^{m-1} f_i(\mathbf{x}).$$

Khi đó  $\rho_m$  sẽ là:

$$\rho_m = \operatorname{argmin}_{\rho} E_{y, \mathbf{x}} L(y, F_{m-1}(\mathbf{x}) - \rho g_m(\mathbf{x}))$$

Như vậy, mục tiêu của chúng ta là đi xây dựng additive model:

$$F(\mathbf{x}; \{\beta_m, \mathbf{a}_m\}_1^M) = \sum_{m=1}^M \beta_m h(x; \mathbf{a}_m)$$

Nhưng sẽ rất khó nếu ta huấn luyện trực tiếp để tìm tập parameters trong không gian tham số:

$$\{\beta_m, \mathbf{a}_m\}_1^M = \operatorname{argmin}_{\{\beta'_m, \mathbf{a}'_m\}} \sum_{i=1}^N L(y_i, \sum_{m=1}^M \beta'_m h(\mathbf{x}_i, \mathbf{a}'_m))$$

Vì vậy, ta sẽ dùng greedy stagewise trong không gian hàm số để giải:

$$(\beta_m, \mathbf{a}_m) = \operatorname{argmin}_{\beta, a} \sum_{i=1}^N L(y_i, F_{m-1}(\mathbf{x}_i) + \beta h(\mathbf{x}_i; a))$$

$$\text{khi đó: } F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta_m h(\mathbf{x}; \mathbf{a}_m)$$

### 3.4 Ridge Classification

Ridge Classification là một phương pháp phân loại rất hiệu quả. Đây là một phương pháp được sử dụng rộng rãi để giải quyết bài toán phân loại trong nhiều lĩnh vực ứng dụng. Với sự kết hợp giữa mô hình hồi quy và hiệu chỉnh, Ridge Classification cho phép chúng ta xây dựng các mô hình phân loại ổn định và khả năng tổng quát hóa tốt. Mô hình này được xây dựng dựa trên cơ sở của Ridge Regression là một biến thể của Linear Regression. Tuy nhiên, Ridge Classification điều chỉnh và mở rộng Ridge Regression để có thể áp dụng trực tiếp cho bài toán phân loại.

- Nhắc lại bài toán hồi quy tuyến tính: Giả định dữ liệu đầu vào bao gồm N quan sát là những cặp các biến đầu vào và biến mục tiêu  $((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_N, y_N))$  Quá trình hồi qui mô hình sẽ tìm kiếm một vec tơ hệ số ước lượng  $\mathbf{w} = [w_0, w_1, \dots, w_p]$  sao cho tối thiểu hoá hàm mất mát dạng MSE:

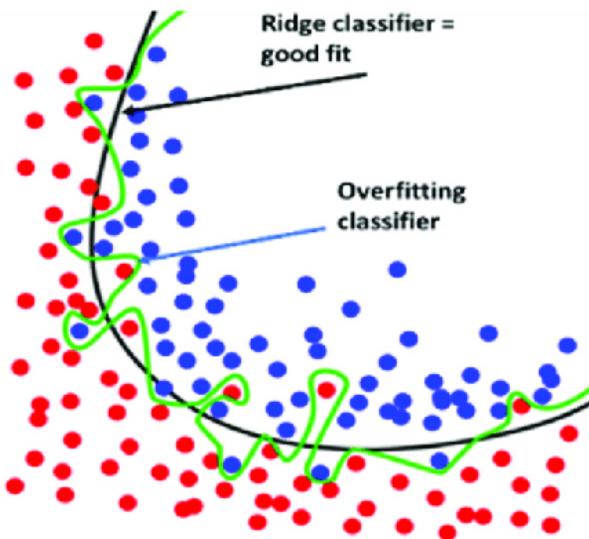
$$\mathcal{L}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 = \frac{1}{N} \|\bar{\mathbf{X}}\mathbf{w} - \mathbf{y}\|_2^2$$

- Khi mô hình tìm được quá phức tạp để mô phỏng training data, nó sẽ dẫn đến hiện tượng quá khớp (Overfitting). Việc quá khớp này có thể dẫn đến việc dự đoán nhầm nhiễu, và chất lượng mô hình không còn tốt trên dữ liệu test nữa. Điều này đặc biệt xảy ra khi lượng dữ liệu training quá nhỏ trong khi độ phức tạp của mô hình quá cao. Một trong những ưu điểm quan trọng của mô hình này là khả năng giảm quá khớp bằng cách thêm thành phần hiệu chỉnh vào hàm mất mát của hồi quy tuyến tính như sau:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \frac{1}{N} \|\bar{\mathbf{X}}\mathbf{w} - \mathbf{y}\|_2^2 + \alpha \|\mathbf{w}\|_2^2 \\ &= \frac{1}{N} \|\bar{\mathbf{X}}\mathbf{w} - \mathbf{y}\|_2^2 + \underbrace{\alpha R(\mathbf{w})}_{\text{regularization term}}\end{aligned}$$

Trong phương trình trên thì  $\alpha \geq 0$ .  $\frac{1}{N} \|\bar{\mathbf{X}}\mathbf{w} - \mathbf{y}\|_2^2$  chính là tổng bình phương phần dư và  $\alpha \|\mathbf{w}\|_2^2$  đại diện cho *thành phần hiệu chỉnh*.

- Bài toán tối ưu hàm mất mát của hồi qui Ridge về bản chất là tối ưu song song hai thành phần bao gồm tổng bình phương phần dư và *thành phần hiệu chỉnh*. Để tìm nghiệm tối ưu cho hàm mất mát, ta có thể sử dụng các solver như saga, lsqr, auto. Hệ số  $\alpha$  có tác dụng điều chỉnh độ lớn của *thành phần hiệu chỉnh* tác động lên mô hình.
  - Trường hợp  $\alpha = 0$ , thành phần hiệu chỉnh bị tiêu giảm và chúng ta quay trở về bài toán hồi qui tuyến tính.
  - Trường hợp  $\alpha$  nhỏ khi ta muốn hiệu chỉnh ít hơn. Mức độ kiểm soát quá khớp của mô hình sẽ trở nên kém hơn.
  - Trường hợp  $\alpha$  lớn khi chúng ta muốn gia tăng mức độ kiểm soát lên độ lớn của các hệ số ước lượng và qua đó giảm bớt hiện tượng quá khớp.



Hình 19: Ridge Classification

Thành phần hiệu chỉnh này giúp kiểm soát sự phức tạp của mô hình và giảm thiểu tác động của nhiễu trong dữ liệu huấn luyện, giúp mô hình có khả năng tổng quát hóa tốt hơn khi dữ liệu có nhiều chiều bằng cách giảm tác động của các đặc trưng không quan trọng.

Mặc dù Ridge Classification có nhiều ưu điểm, nhưng nó cũng có một số hạn chế cần được lưu ý. Ví dụ, Ridge Classification có tính tương đối, nghĩa là mô hình có thể không phân loại chính xác các điểm dữ liệu gần biên của các lớp như hình 30d. Điều này có thể làm giảm độ chính xác và độ tin cậy của mô hình trong những tình huống gần biên.

## 4 Kết quả thực nghiệm

Phần này của báo cáo sẽ tiến hành thực nghiệm để kiểm tra kết quả dự đoán của bốn 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 được xử lý khác nhau, 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. Nhóm sẽ so sánh độ chính xác của các mô hình với những các xử lý bộ dữ liệu khác nhau. 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.

Bộ dữ liệu được chia làm hai cách xử lý: để nguyên màu và làm mất màu bằng lapcian. Các mô hình sẽ được huấn luyện trên tập dữ liệu có màu, không màu rồi đem đi so sánh độ chính xác của các mô hình trên bộ dữ liệu kiểm tra với 2 cách xử lý khác nhau.

Sau khi trải qua các bước xử lý và trích xuất đặc trưng, tập dữ liệu được coi như dữ liệu dạng bảng với mỗi ảnh có 1000 thuộc tính.

Phần thực nghiệm sẽ tập trung phân tích vào độ hiệu quả của mô hình khi dự đoán hoa bồ công anh, độ đo f1-score sẽ được dùng để đánh giá mô hình do dữ liệu cân bằng và mức độ quan trọng của 8 loài hoa là như nhau.

## 4.1 Mô hình huấn luyện trên bộ có màu và dự đoán tập ảnh có màu

### 4.1.1 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 : 5, 6, 7, 8, 9 , 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:

$$Weight1 = distances$$

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

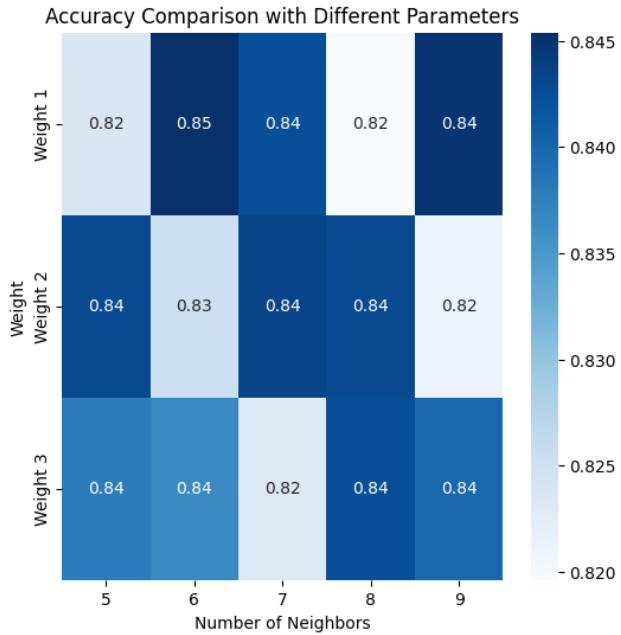
$$Weight3 = \frac{1}{distances^2}$$

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

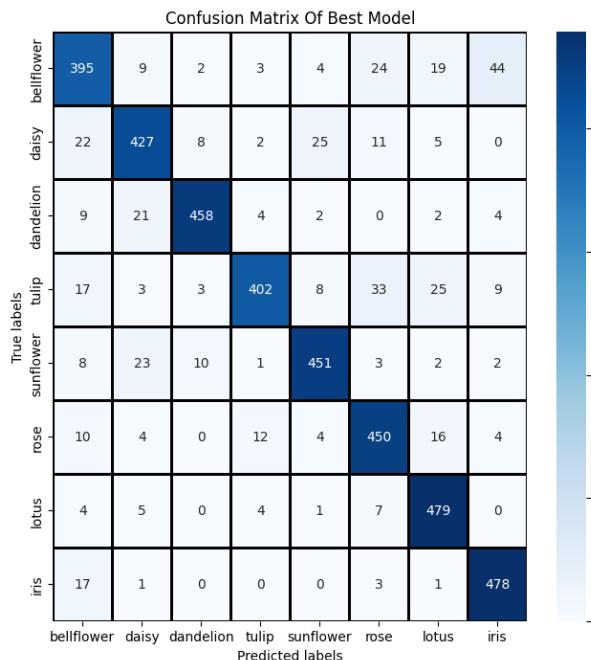
Tương ứng với mỗi tham số trên, mô hình sẽ lần lượt được huấn luyện với tập dữ liệu validation các bộ dữ liệu chưa được chuẩn hoá và đã được chuẩn hoá.

Độ hiệu quả của mô hình KNN trên tập dữ liệu xác thực kết hợp với các tham số như: số hàng xóm, cách tính trọng số,... được thể hiện trong hình ??.

Với tham số tốt nhất được chọn là 6 hàng xóm, hàm khoảng cách weight 1, mô hình KNN cho hiệu quả trên tập xác thực được thể hiện trong lần lượt các hình [20b](#), [20c](#). Có thể thấy rõ, mô hình dự đoán tốt trên tập xác thực với sự nhầm lẫn giữa các hoa lẫn nhau là rất ít với tỷ lệ chính xác 0.88, giá trị f1-score lớn nhất với hoa bồ công anh và thấp nhất với hoa chuông. Hoa chuông trong mô hình knn bị dự đoán sai chủ yếu thành hoa diên vĩ.



(a) Hiệu chỉnh trên tập xác thực



(b) Confusion matrix

	precision	recall	f1-score
bellflower	0.819502	0.790	0.804481
daisy	0.866126	0.854	0.860020
dandelion	0.952183	0.916	0.933741
tulip	0.939252	0.804	0.866379
sunflower	0.911111	0.902	0.906533
rose	0.847458	0.900	0.872939
lotus	0.872495	0.958	0.913251
iris	0.883549	0.956	0.918348
accuracy	0.885000	0.885	0.885000

(c) Các độ đo đánh giá

Hình 20: Hiệu chỉnh tham số rồi kiểm thử trên tập kiểm tra

#### 4.1.2 eXtreme Gradient Boosting (Xgboost)

Mô hình Xgboost để giải bài toán phân loại hoa sử dụng thuật toán XGBClassifier. 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.

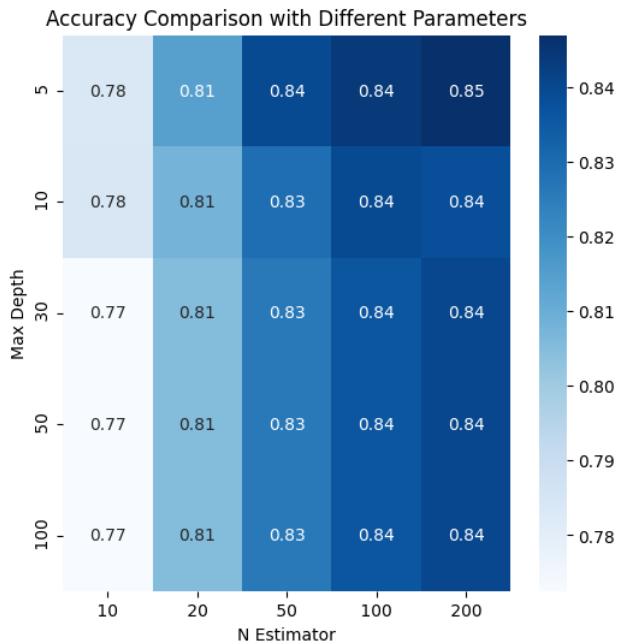
Trong quá trình thực nghiệm chúng tôi thấy rằng khả năng dự đoán trên tập xác thực thì số thuộc tính được chọn để đánh giá và chia nhánh dữ liệu có sự ngẫu nhiên cao sẽ cho kết quả không chênh lệch hơn với các thuộc tính cho sự ngẫu nhiên thấp,hình 21a cho ta thấy với số lượng cây càng lớn, tỷ lệ chính xác của mô hình sẽ càng tăng. Khi số cây lớn, mô hình sẽ có xu hướng tăng độ chính xác khi số thuộc tính dùng để nhánh giảm, điều này cho thấy sự cân bằng giữa số lượng cây và thuộc tính để đảm bảo mô hình không học vẹt với tập dữ liệu. Với số lượng thuộc tính của bộ dữ liệu dùng để phân loại (1000 thuộc tính) thì số lượng thuộc tính được chọn để đánh giá tốt nhất là 5 với 200 cây để chia nhánh dữ liệu.

Hình 21b và 21c cho ta thấy với tham số tốt nhất, khả năng dự đoán của mô hình Xgboost rất tốt. Hoa bồ công anh và hoa chuông vẫn có giá trị f1-score cao nhất và thấp nhất. So sánh với 22, gồm 10 cây và 5 thuộc tính (rất ít so với max\_features=1000) dùng để chia nhánh dữ liệu thì kết quả chênh lệch không đáng kể.

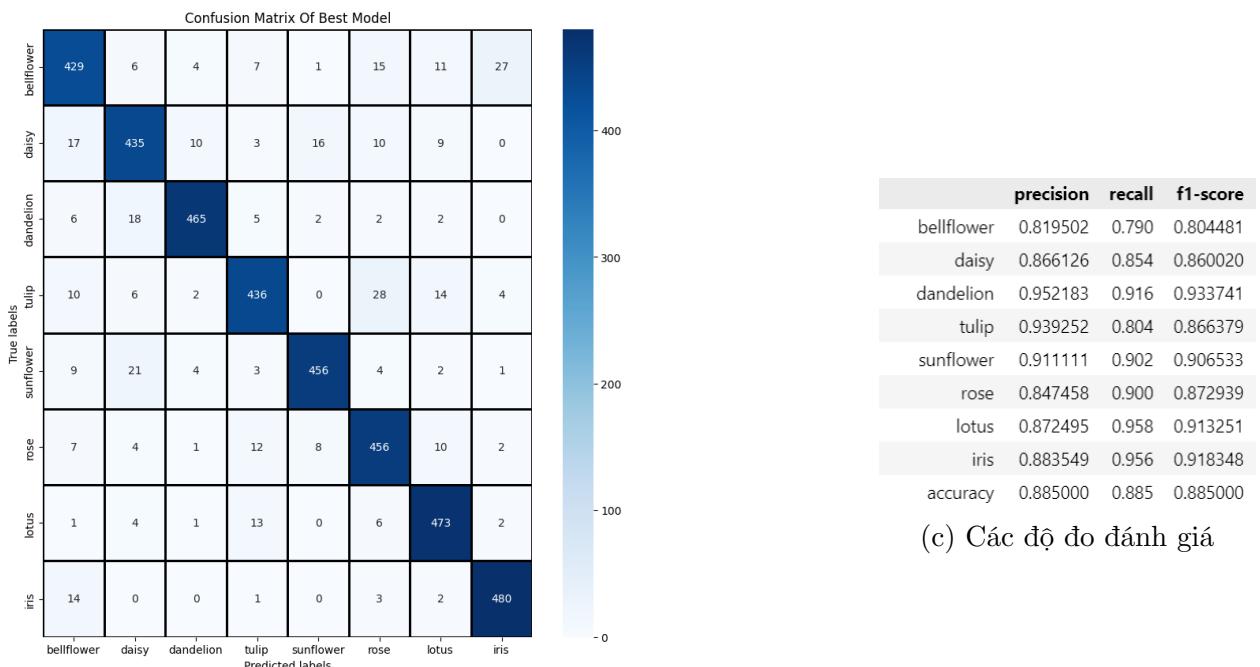
	precision	recall	f1-score
bellflower	0.739394	0.7320	0.735678
daisy	0.858065	0.7980	0.826943
dandelion	0.902584	0.9080	0.905284
tulip	0.826613	0.8200	0.823293
sunflower	0.878296	0.8660	0.872105
rose	0.819367	0.8800	0.848602
lotus	0.859213	0.8300	0.844354
iris	0.820076	0.8660	0.842412
accuracy	0.837500	0.8375	0.837500

Hình 22: Các độ đo đánh giá mô hình trên tập kiểm tra ứng sử dụng 5 thuộc tính để chia nhánh dữ liệu với số lượng cây 10.

Điều này có thể do có những thuộc tính mang thông tin quan trọng hơn những thuộc tính còn lại, như thuộc tính về hoa mang nhiều thông tin và có giá trị cao hơn những thuộc tính về môi trường. Việc chia nhánh dữ liệu bằng những thuộc tính này giúp mô hình học hiệu quả cho trong trường hợp số cây và số thuộc tính để chia nhánh ít



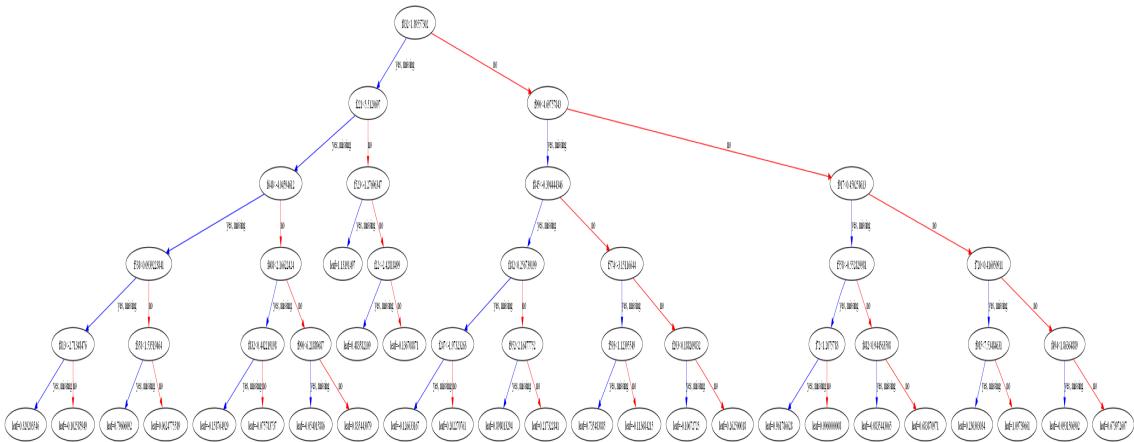
(a) Hiệu chỉnh trên tập xác thực



(c) Các độ đo đánh giá

(b) Confusion matrix

Hình 21: Hiệu chỉnh tham số rồi kiểm thử trên tập kiểm tra



Hình 23: Một cây trong mô hình XGboost.

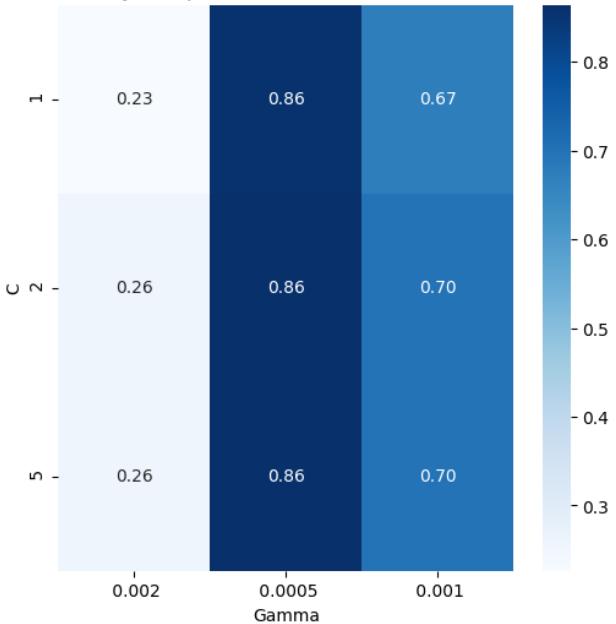
#### 4.1.3 Support Vector Machine (SVM)

Trong phần thử nghiệm mô hình SVM, thuật toán Kernel support vector machine (Kernel SVM) sẽ được áp dụng để phân loại, hàm Kernel được sử dụng là Radial basic function (RBF) vì dữ liệu không biết trước cấu trúc. Mô hình SVM sẽ thử nghiệm với tham số C: 1, 2, 5 và gamma: 0.002, 0.0005, 0.001.

Hình 24a cho thấy sự ảnh hưởng khi thay đổi tham số C không quá đáng kể ngược lại tham số gamma ảnh hưởng rất lớn đối với khả năng dự đoán chính xác của mô hình. Khi gamma tăng, độ chính xác của mô hình giảm, điều này có thể do dữ liệu mang nhiều thuộc tính nhiễu: môi trường xung quanh... hoặc do chúng ta không thể quan sát trước được cấu trúc của dữ liệu.

Với tham số gamma được chọn chính xác, mô hình tỏ ra hiệu quả với tỷ lệ dự đoán chính xác cao. Hoa bồ công anh vẫn có giá trị f1-score cao chỉ đứng sau hoa diên vĩ. Hoa chuông có giá trị f1-score thấp nhất và trong các trường hợp phán đoán sai thường bị nhầm sang hoa diên vĩ.

Accuracy Comparison with Different Parameters



(a) Hiệu chỉnh trên tập xác thực

Confusion Matrix Of Best Model

True labels	Predicted labels							
	bellflower	daisy	dandelion	tulip	sunflower	rose	lotus	iris
bellflower	437	4	2	8	1	16	11	21
daisy	14	440	8	7	9	17	5	0
dandelion	9	9	470	7	0	2	3	0
tulip	10	1	1	448	0	23	12	5
sunflower	6	14	6	11	456	5	2	0
rose	5	4	0	13	1	469	6	2
lotus	1	3	1	9	0	5	480	1
iris	6	0	0	1	0	1	2	490

(b) Confusion matrix

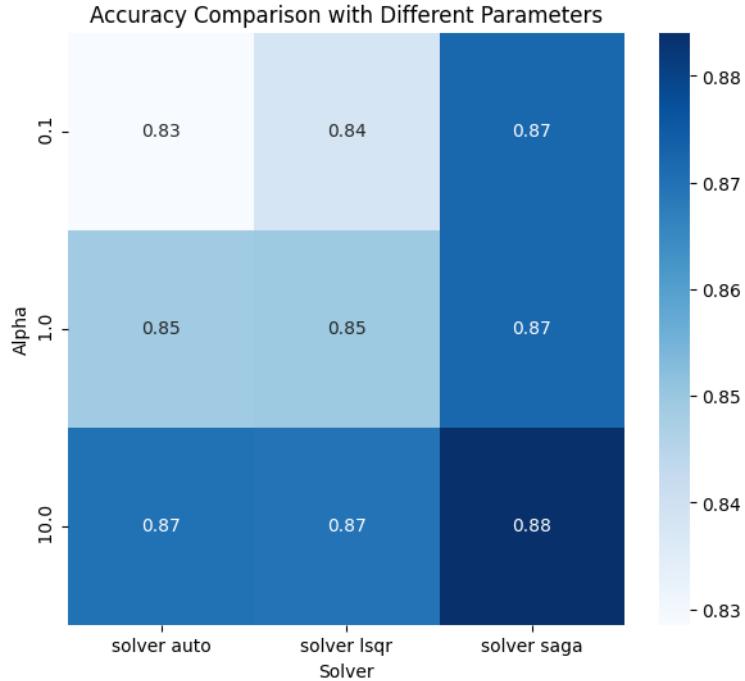
	precision	recall	f1-score
bellflower	0.895492	0.8740	0.884615
daisy	0.926316	0.8800	0.902564
dandelion	0.963115	0.9400	0.951417
tulip	0.888889	0.8960	0.892430
sunflower	0.976445	0.9120	0.943123
rose	0.871747	0.9380	0.903661
lotus	0.921305	0.9600	0.940255
iris	0.944123	0.9800	0.961727
accuracy	0.922500	0.9225	0.922500

(c) Các độ đo đánh giá

Hình 24: Hiệu chỉnh tham số rồi kiểm thử trên tập kiểm tra

#### 4.1.4 Ridge classification

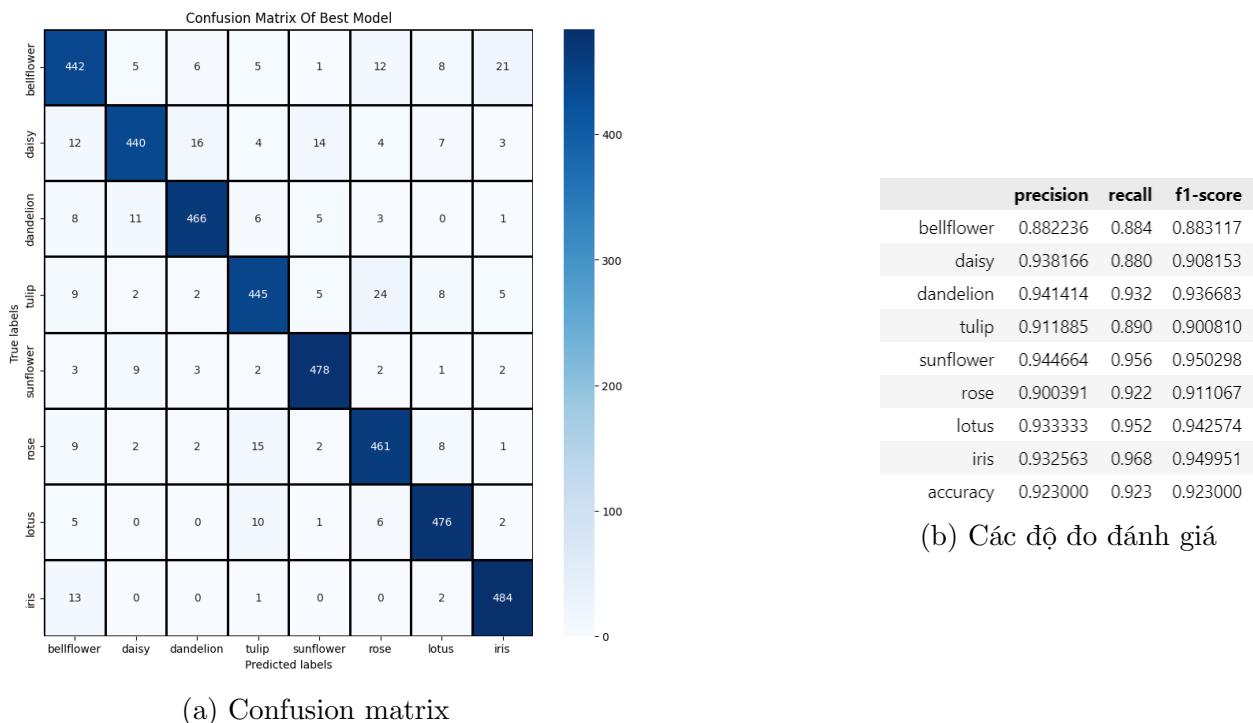
Trong phần này, mô hình ridge classification sẽ được kiểm tra độ hiệu quả đạt được khi thay đổi các tham số liên quan đến cách mô hình học và trọng số *alpha* của regularization.



Hình 25: Hiệu chỉnh tham số trên tập xác thực

Theo quan sát trong hình 25, *saga* solver có độ chính xác cao nhất, tham số alpha tốt nhất được chọn là 10. Với alpha càng nhỏ, khả năng dự đoán của mô hình càng kém. Nguyên nhân có thể do có nhiều thuộc tính nhiễu khiến khi tham số alpha nhỏ, regularization càng gần 0 khiến mô hình bị ảnh hưởng lớn bởi các thông tin nhiễu.

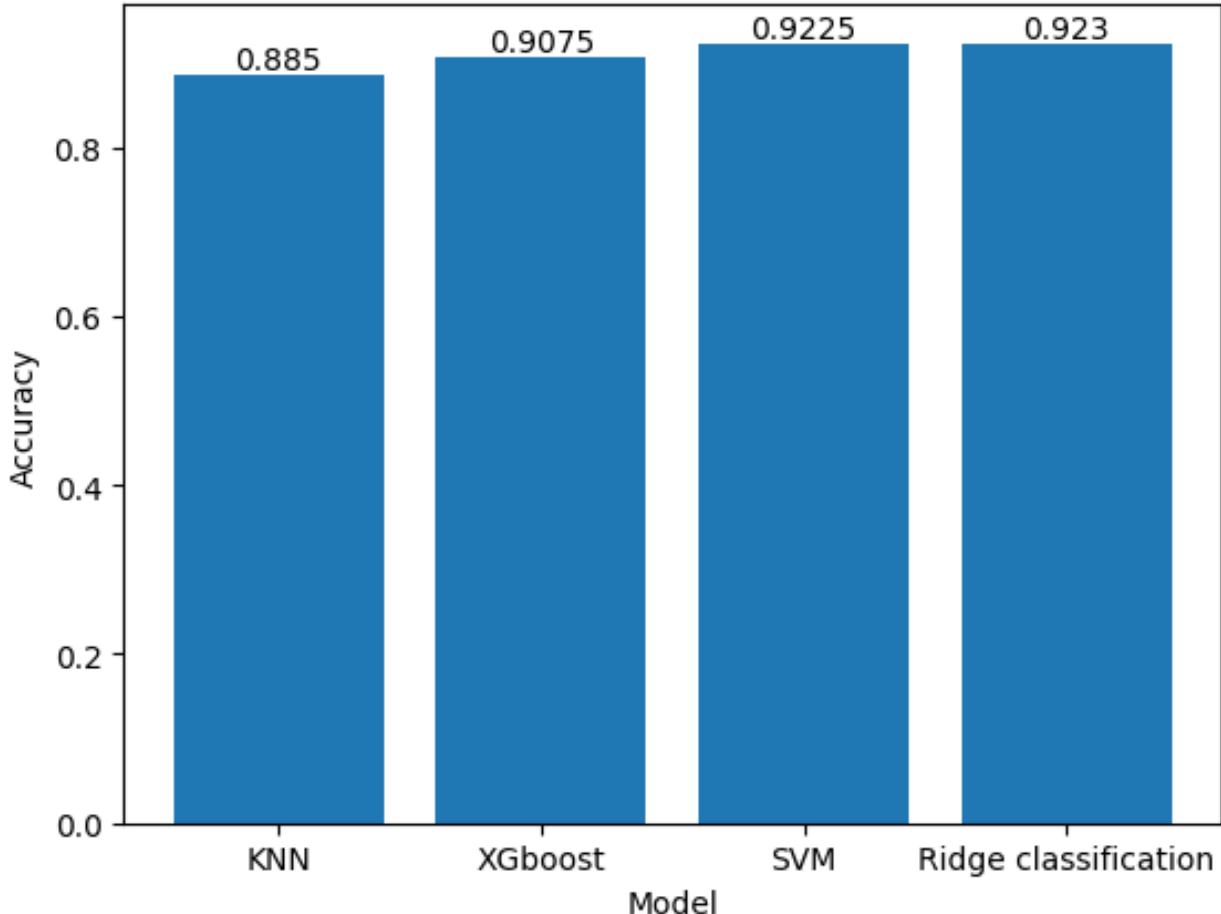
Hình 26a và 31b không cho thấy nhiều sự khác biệt với các mô hình trước, hoa chuông vẫn có độ chính xác thấp nhất và hay bị nhầm sang hoa diên vĩ trong các trường hợp dự đoán sai. Hoa bồ công anh vẫn có giá trị f1-score cao và hay bị hoa cúc dự đoán nhầm thành.

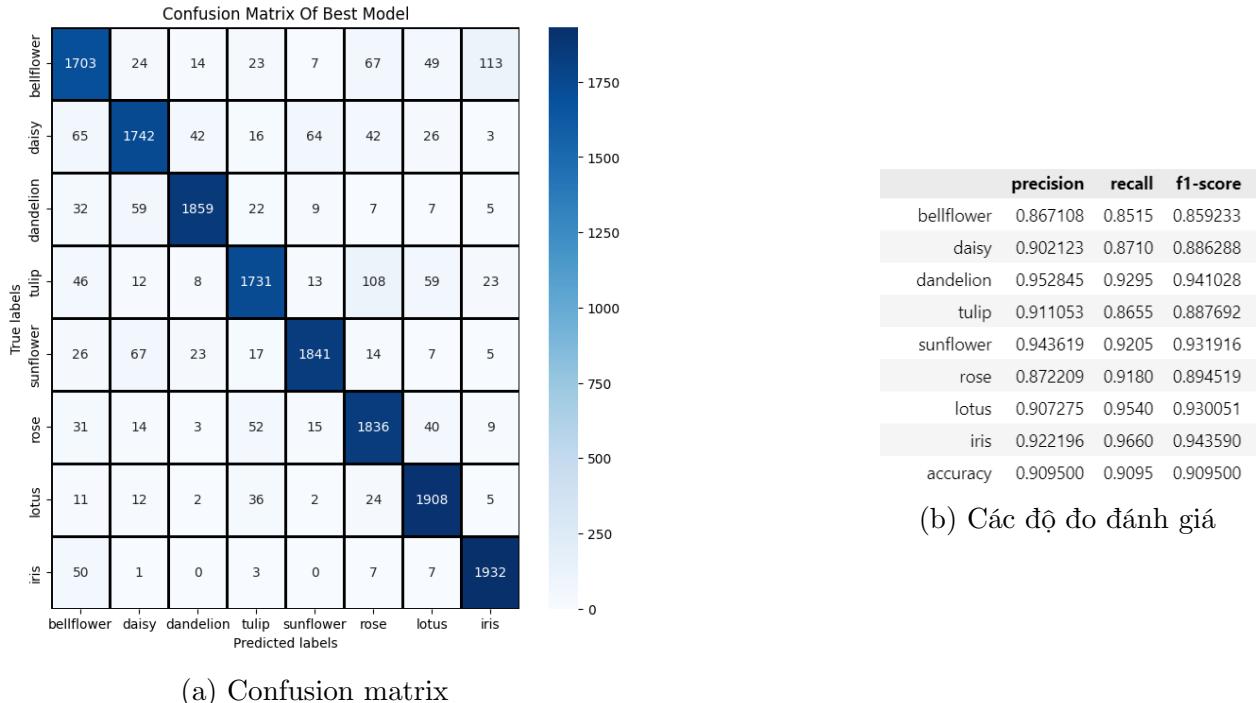


(a) Confusion matrix

Hình 26: Hiệu chỉnh tham số rồi kiểm thử trên tập kiểm tra

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



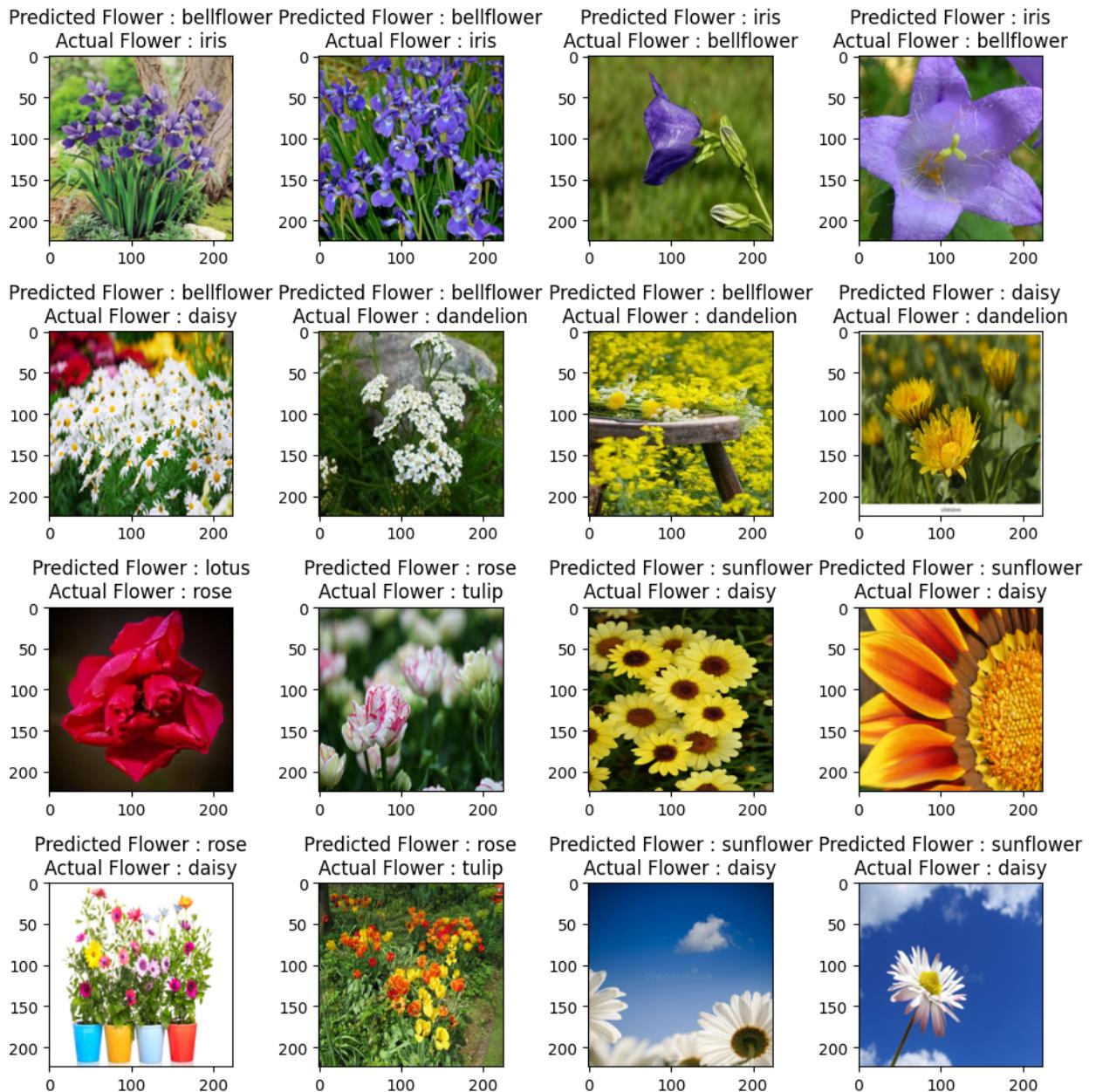


Hình 28: Đánh giá tổng hợp các mô hình trên tập kiểm tra.

Sau khi lựa chọn được các tham số để lại kết quả tốt nhất ở những bước thực nghiệm trước, các mô hình sẽ được đếm dự đoán các loài hoa trên bộ dữ liệu kiểm tra. Độ chính xác được biểu diễn trong hình 27

Nhìn chung các mô hình sử dụng siêu phẳng để phân chia dữ liệu như SVM và Ridge classification có hiệu quả dự đoán chính xác cao nhất, thấp nhất là KNN. Nguyên nhân có thể là do KNN có khả năng tổng quát hóa kém hơn với bộ dữ liệu lớn và có chứa nhiều nhiễu. Mô hình XGboost có độ chính xác thấp hơn do với mô hình rừng sử dụng các cây để chia nhánh dữ liệu, mô hình sẽ không học được nhiều những tri thức về mối liên hệ giữa các thuộc tính với nhau.

Tổng hợp lại dữ liệu đánh giá từ các mô hình, hình 28a và 28 cho thấy hoa bồ công anh có giá trị f1-score chỉ đứng sau hoa diên vĩ, hoa chuông có giá trị f1-score thấp nhất. Từ confusion matrix ta thấy được hoa chuông chủ yếu bị dự đoán nhầm sang hoa diên vĩ và ngược lại phần lớn hoa diên vĩ bị nhầm sang hoa chuông. Hoa bồ công anh, hoa cúc và hoa hướng dương, giữa chúng thường bị các mô hình dự đoán nhầm lẫn.



Hình 29: Trực quan hóa những ảnh bị dự đoán sai.

Hình 29 cho thấy theo trực quan, hoa diêm vĩ và hoa chuông giũa chúng có sự tương đồng cao về màu sắc. Hoa hướng dương, hoa bồ công anh trong giai đoạn phát triển có và hoa cúc có hình dáng tương đối giống nhau, hoa cúc và hoa bồ công anh có cùng 2 loại màu sắc: trắng và vàng. Trong trường hợp hoa cúc vàng và hoa bồ công anh đang phát triển, dựa vào trực quan khó nhận ra điểm khác biệt với hoa hướng dương. Sự khó phân biệt giữa các dữ liệu khiến cho mô hình phán đoán không chính xác.

## 4.2 Các mô hình được huấn luyện và phán đoán trên tập dữ liệu không màu

Trong phần này, các mô hình đều được huấn luyện và đánh giá theo các bước trong phần trên. Sự khác biệt ở đây là tất cả các ảnh trong tập huấn luyện, xác thực và kiểm tra đều được làm mờ màu trước khi trích xuất đặc tính.

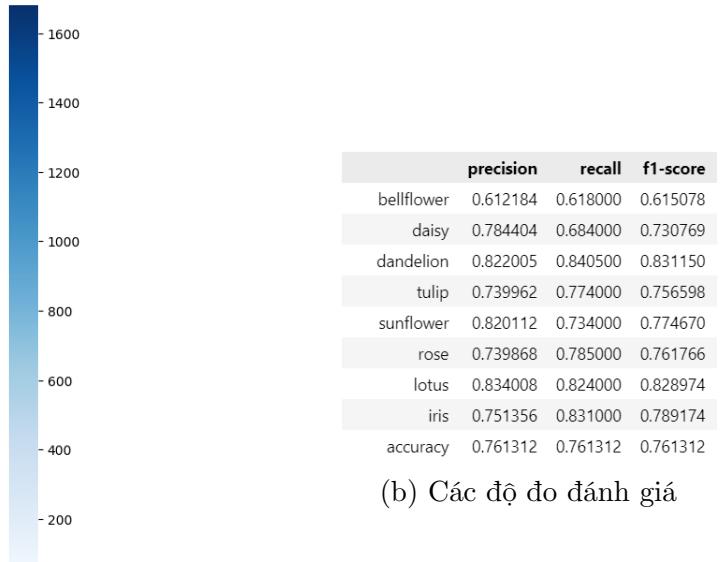
	<b>precision</b>	<b>recall</b>	<b>f1-score</b>		<b>precision</b>	<b>recall</b>	<b>f1-score</b>	
bellflower	0.524109	0.5000	0.511771		bellflower	0.623077	0.64800	0.635294
daisy	0.778589	0.6400	0.702525		daisy	0.764434	0.66200	0.709539
dandelion	0.784762	0.8240	0.803902		dandelion	0.812016	0.83800	0.824803
tulip	0.689394	0.7280	0.708171		tulip	0.752988	0.75600	0.754491
sunflower	0.795122	0.6520	0.716484		sunflower	0.807692	0.75600	0.780992
rose	0.676678	0.7660	0.718574		rose	0.743833	0.78400	0.763389
lotus	0.778218	0.7860	0.782090		lotus	0.852697	0.82200	0.837067
iris	0.660900	0.7640	0.708720		iris	0.760870	0.84000	0.798479
accuracy	0.707500	0.7075	0.707500		accuracy	0.763250	0.76325	0.763250
(a) KNN				(b) XGboost				
	<b>precision</b>	<b>recall</b>	<b>f1-score</b>		<b>precision</b>	<b>recall</b>	<b>f1-score</b>	
bellflower	0.667969	0.684	0.675889		bellflower	0.627451	0.6400	0.633663
daisy	0.806522	0.742	0.772917		daisy	0.786364	0.6920	0.736170
dandelion	0.868421	0.858	0.863179		dandelion	0.825490	0.8420	0.833663
tulip	0.791506	0.820	0.805501		tulip	0.727941	0.7920	0.758621
sunflower	0.839479	0.774	0.805411		sunflower	0.835920	0.7540	0.792850
rose	0.761993	0.826	0.792706		rose	0.784394	0.7640	0.774063
lotus	0.887295	0.866	0.876518		lotus	0.820359	0.8220	0.821179
iris	0.828571	0.870	0.848780		iris	0.763016	0.8500	0.804163
accuracy	0.805000	0.805	0.805000		accuracy	0.769500	0.7695	0.769500
(c) SVM				(d) Ridge classification				

Hình 30: Đánh giá tổng hợp các mô hình trên tập kiểm tra.

Hình 30 cho thấy mô hình SVM có khả năng dự đoán chính xác cao nhất, mô hình KNN dự đoán tệ nhất trong 4 mô hình, 2 mô hình XGboost và Ridge Classification có kết quả dự đoán khá tương đồng nhau. Trong cả 4 mô hình, hoa chuông vẫn có kết quả dự đoán kém nhất, hoa bồ công anh trong các mô hình được dự đoán với độ chính xác cao, ổn định.

Confusion Matrix Of Best Model								
	bellflower	daisy	dandelion	tulip	sunflower	rose	lotus	iris
True labels	1236	55	64	109	37	189	85	225
bellflower	167	1368	87	53	145	60	72	48
daisy	103	69	1681	37	52	8	14	36
dandelion	108	31	33	1548	39	110	63	68
tulip	152	141	88	56	1468	36	24	35
sunflower	118	45	32	95	22	1570	34	84
rose	50	28	26	113	12	69	1648	54
lotus	85	7	34	81	15	80	36	1662
iris								
Predicted labels	bellflower	daisy	dandelion	tulip	sunflower	rose	lotus	iris

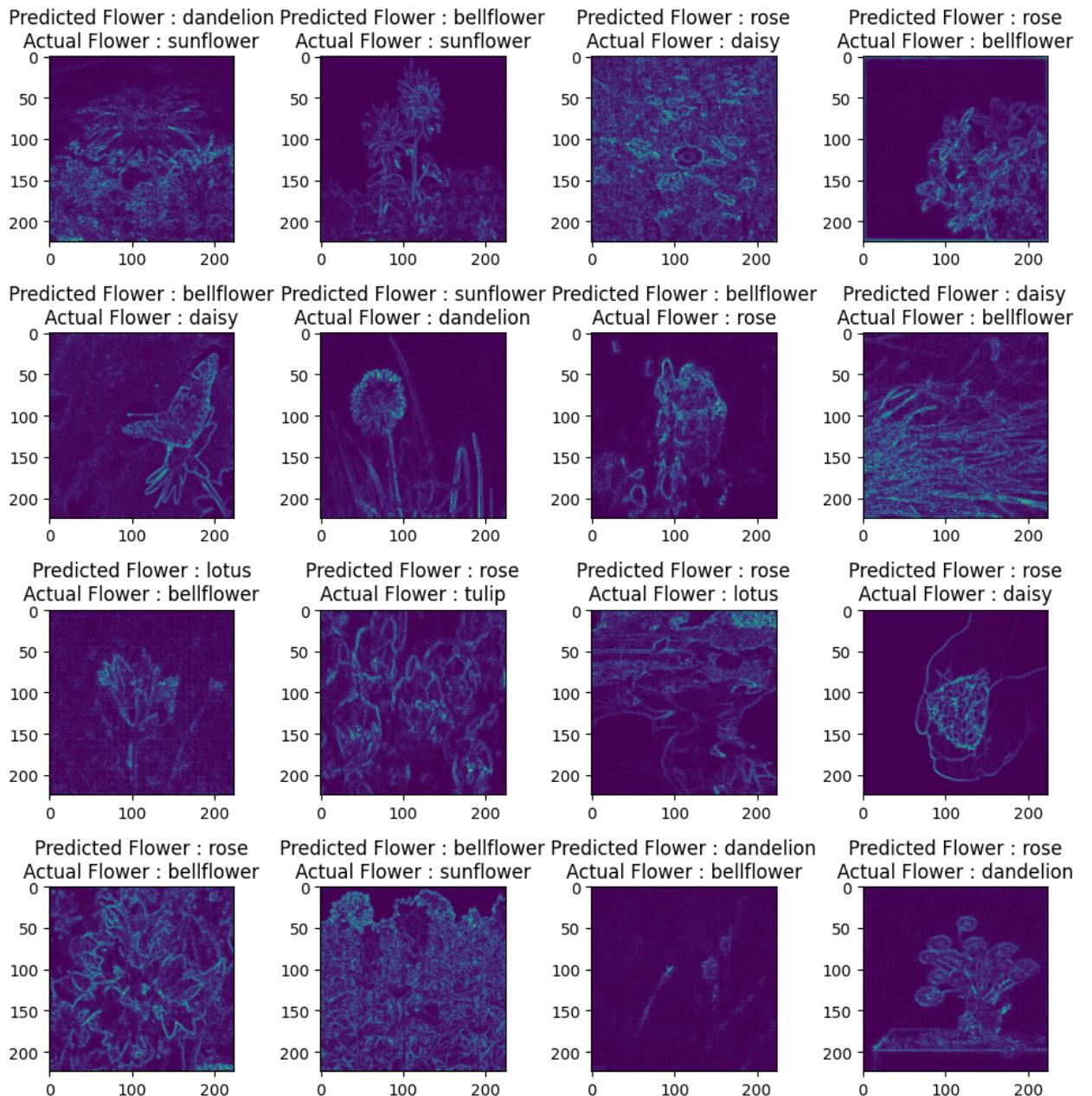
(a) Confusion matrix



(b) Các độ đo đánh giá

Hình 31: Đánh giá tổng hợp các mô hình trên tập kiểm tra.

Dựa vào hình 31a và 31, ta thấy khả năng dự đoán của các mô hình giảm đi đáng kể so với việc học trên ảnh có màu rồi dự đoán ảnh không màu. Phần lớn các mô hình dự đoán nhầm các loài hoa khác sang hoa chuông và ngược lại hoa chuông cũng bị dự đoán nhầm sang các hoa khác rất nhiều, chủ yếu là hoa hồng, tulip và hoa diên vĩ.



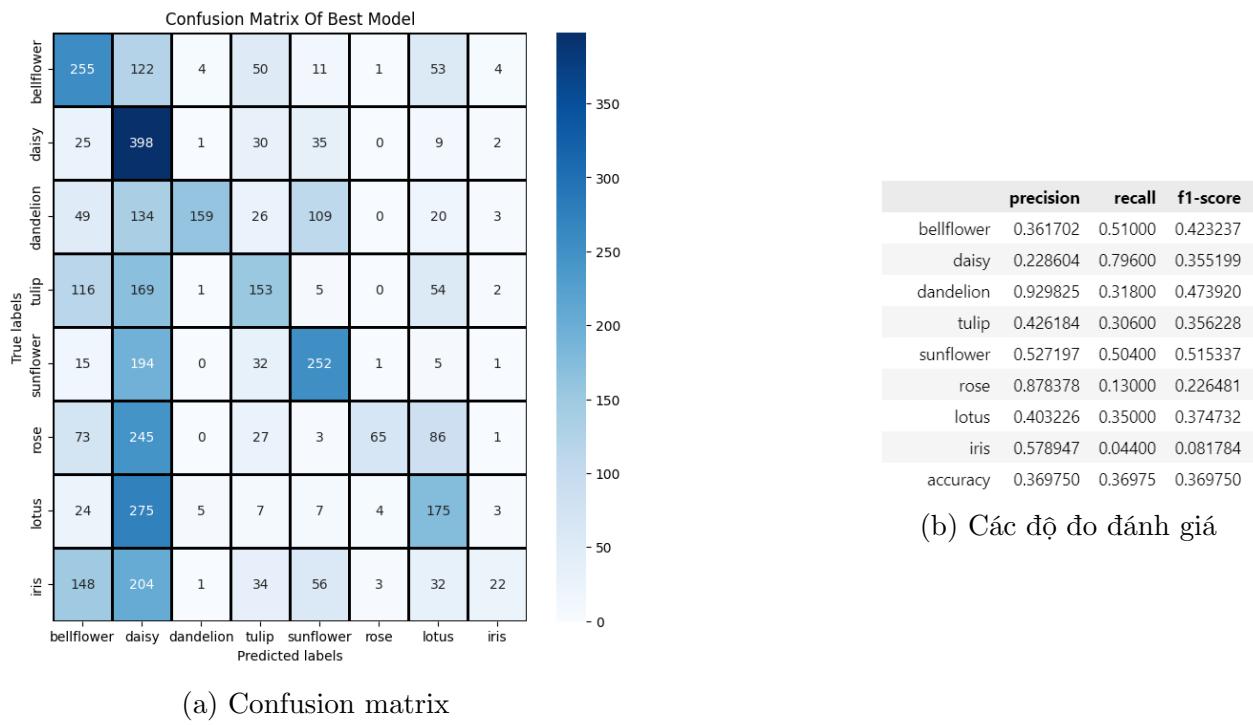
Hình 32: Những ảnh bị phán đoán sai.

Với việc nhiều loài hoa có cấu trúc khá tương đồng, việc thiếu thông tin về màu sắc làm cho mô hình học khó phân biệt được các loài hoa khiến cho kết quả phán đoán tệ đi. Với việc thiếu thông tin về màu sắc như trong hình 32, hoa bồ công anh, hoa hướng dương và hoa cúc rất khó để phân biệt lẫn nhau; hoa sen khi chưa xòe cánh hoa, hoa chuông và hoa tulip có hình dáng tương đối giống nhau. Có thể thấy rằng thông tin về màu sắc ảnh hưởng một phần không nhỏ đến khả phán đoán của các mô hình.

### 4.3 Mô hình học trên tập dữ liệu không màu và phán đoán trên tập có màu

Phần này nhóm muốn đánh giá các mô hình nếu được học trên tập dữ liệu thiếu thông tin về màu sắc thì khả năng phán đoán với tập dữ liệu có thông tin về màu sắc có hiệu quả như thế nào.

#### 4.3.1 KNN

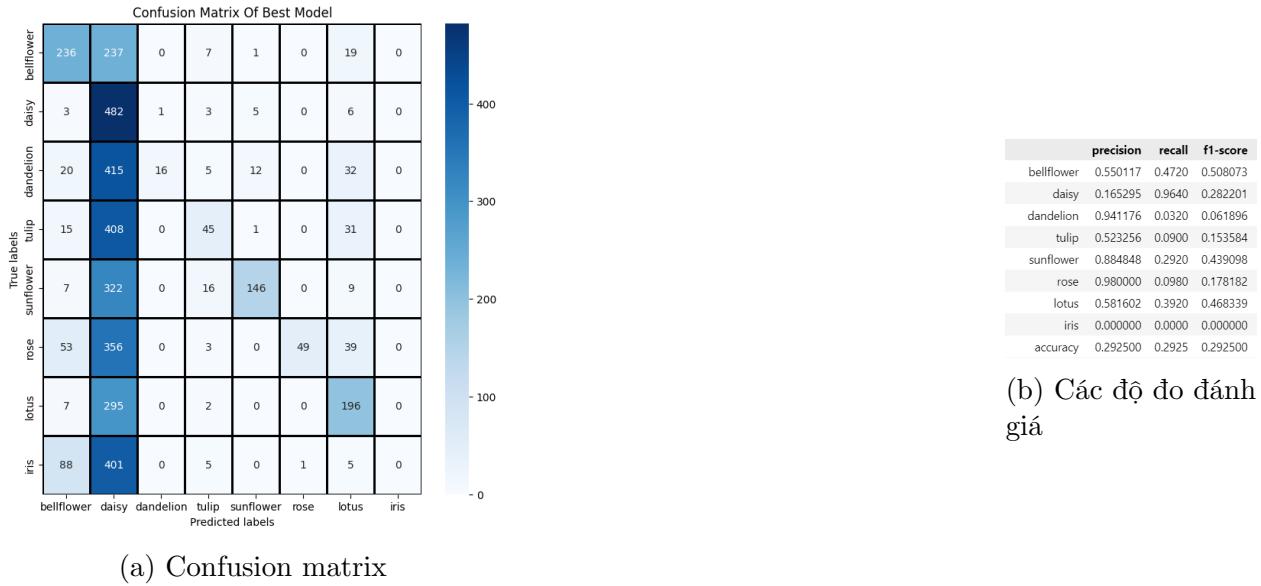


Hình 33: Đánh giá tổng hợp các mô hình trên tập kiểm tra.

Hình 33a và 33b cho thấy mô hình chủ yếu dự đoán lệch sang hoa cúc với giá trị recall cao nhất, hoa bồ công anh và hoa diên vĩ hầu như không có loài hoa nào bị phán đoán nhầm sang nhưng khả năng phán đoán chính xác của hoa diên vĩ lại rất thấp. Mô hình khô

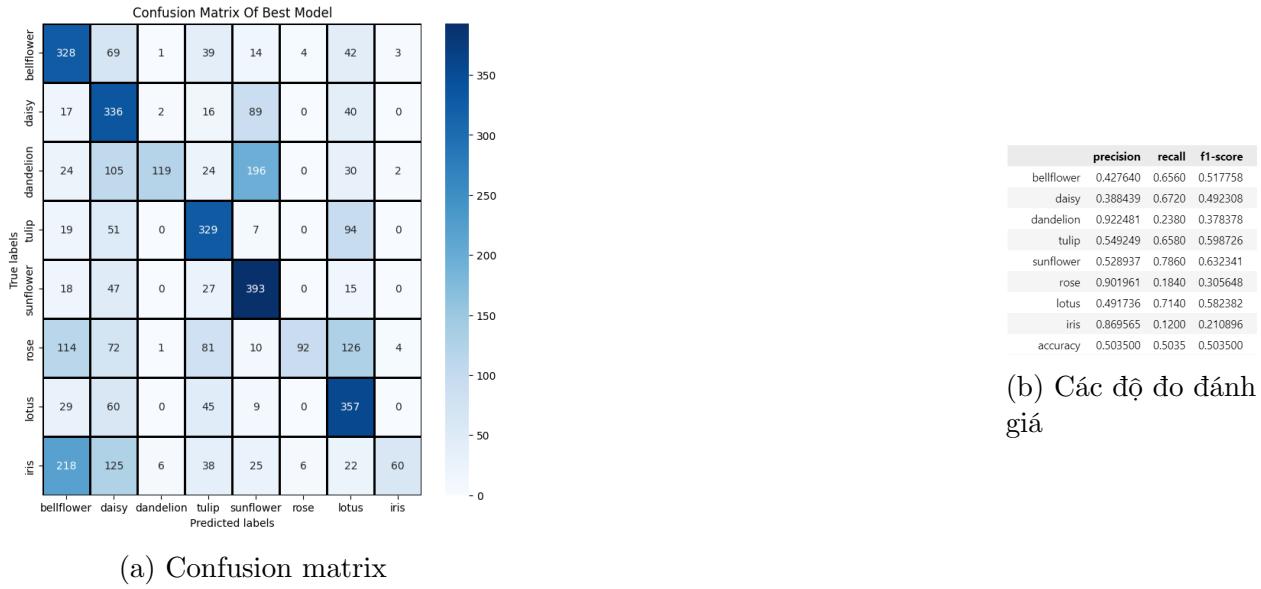
#### 4.3.2 SVM

Dựa vào hình 34a và 34b mô hình có tỷ lệ cao dự đoán nhầm sang hoa cúc, khả năng nhận ra các loài hoa khác là rất thấp. Mô hình có thể đang gặp vấn đề overfitting (quá khớp) trên tập dữ liệu không màu và không tổng quát hóa tốt cho tập dữ liệu có màu khi không học được các đặc trưng màu sắc quan trọng trong quá trình phân loại hoa.



Hình 34: Dánh giá tổng hợp các mô hình trên tập kiểm tra.

#### 4.3.3 XGboost

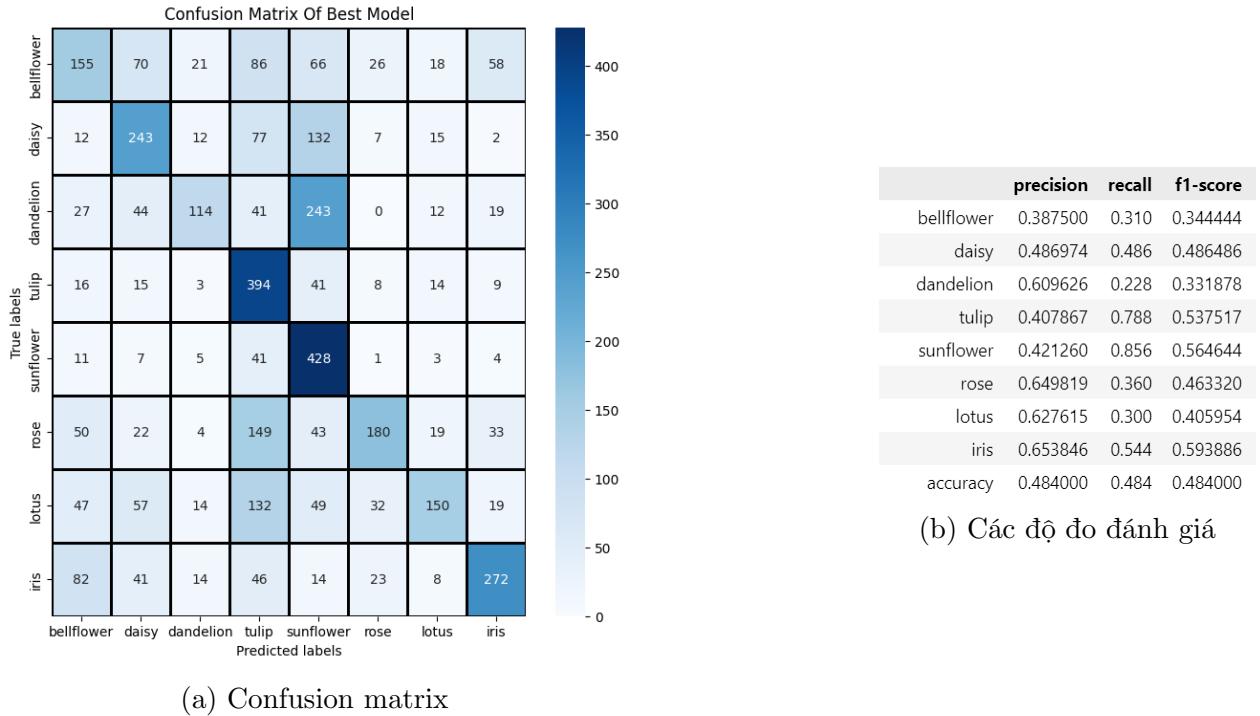


Hình 35: Dánh giá tổng hợp các mô hình trên tập kiểm tra.

Hình 35a và 35b cho thấy mô hình vẫn có thiên hướng dự đoán nhầm sang hoa cúc nhưng không bị lệch quá nhiều, hoa diên vĩ chủ yếu bị phán đoán sai thành hoa chuông. Khả năng mô hình nhận diện ra hoa bồ công anh và hoa hồng thấp với recall: 0.23, 0.18.

#### 4.3.4 Ridge classification

Kết quả dự đoán theo hình 36a và 36b cho thấy mô hình không bị chủ yếu dự đoán nhầm sang 1 loài hoa nào, hoa bồ công anh có giá trị recall thấp nhất cho thấy mô hình khó



(a) Confusion matrix

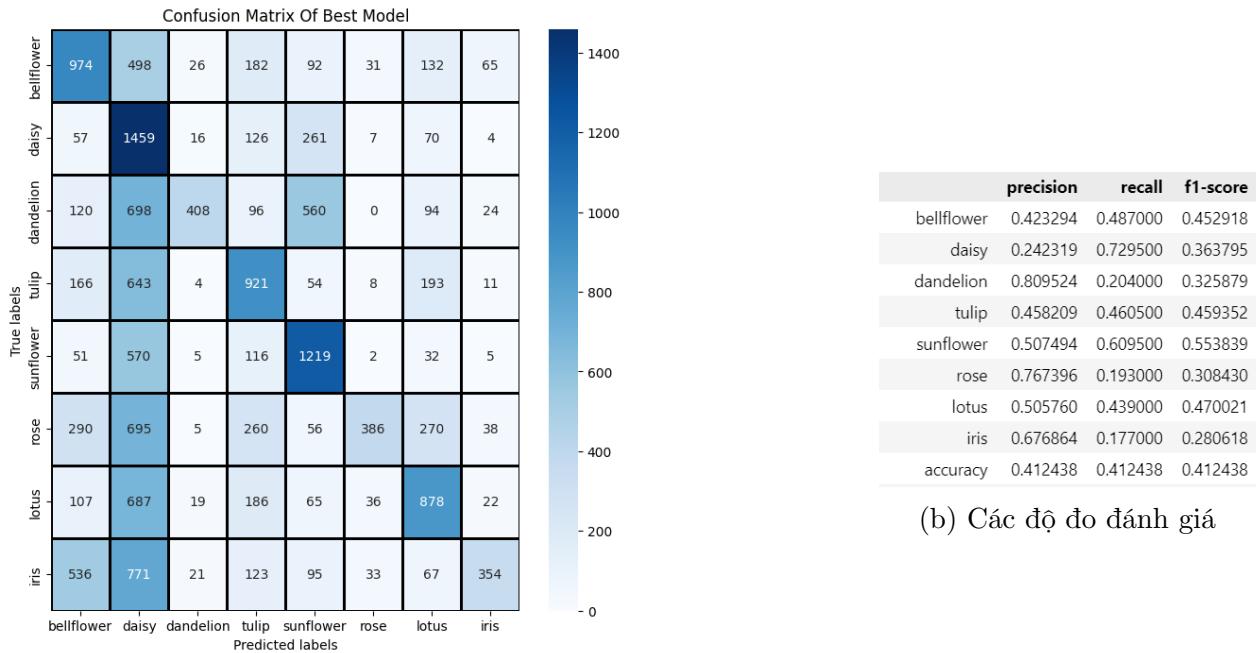
Hình 36: Đánh giá tổng hợp các mô hình trên tập kiểm tra.

nhận ra được.

#### 4.3.5 Tổng quan kết quả phán đoán của các mô hình

Có thể thấy khả năng phán đoán trên tập kiểm tra là ảnh có màu chỉ bằng dữ liệu huấn luyện không mang thông tin về màu sắc là tương đối khó như hình 37b với độ chính xác trong phán đoán 0.41.

Theo hình 37a hầu hết ảnh bị các mô hình phán đoán nhầm sang hoa cúc với giá trị precision là thấp nhất: 0.24; hoa bồ công anh, hoa tulip và hoa diên vĩ có khả năng nhận diện chính xác là không cao, trái ngược lại hoàn toàn khi đưa các mô hình học trên tập huấn luyện có màu. Điều này có thể giải thích là do cấu trúc của hoa cúc rất tương đồng với các loài hoa khác như hoa bồ công anh, hoa hướng dương khi không có màu sắc. Việc này khiến cho các mô hình học lệch và dự đoán sai khi chúng không có tri thức về thông tin về màu sắc khi phán đoán trong tập kiểm tra.



(a) Confusion matrix

Hình 37: Đánh giá tổng hợp các mô hình trên tập kiểm tra.

## 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](https://github.com/Tahuubinh/ML_HUST_Project)

## 6 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] M. Shaha and M. Pawar, *Transfer Learning for Image Classification*, 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2018, pp. 656-660, doi: 10.1109/ICECA.2018.8474802.
- [4] R. M. Haralick, K. Shanmugam and I. Dinstein, *Textural features for image classification*. IEEE Transactions on Systems Man and Cybernetics, vol. SMC-3, no. 6, pp. 610-621, Nov 1973.
- [5] J. D. A. Berg and L. Fei-Fei, *Large scale visual recognition challenge*, 2010, 2010, [online] Available: <http://image-net.org/download>.
- [6]