ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN TỬ - VIỄN THÔNG

BÁO CÁO

TRÍ TUỆ NHÂN TẠO

Bài tập 1: Xây dựng hệ thống AI đánh giá độ ngọt của xoài qua màu sắc vỏ

Nhóm 10: Trần Đăng Bách

Trịnh Anh Nhật

Nguyễn Văn Quý

Lớp: 22KTMT

1. Giới thiệu

1.1. Giới thiệu bài toán

Trong những năm gần đây, việc ứng dụng xử lý ảnh và trí tuệ nhân tạo vào nông nghiệp ngày càng được quan tâm, đặc biệt trong việc đánh giá chất lượng nông sản. Một trong những bài toán thực tế là xác định độ chín của quả xoài, từ đó suy ra độ ngọt hoặc độ chua, nhằm hỗ trợ phân loại, thu hoạch và phân phối sản phẩm.

Thông thường, xoài chín (ngọt) có vỏ màu vàng hoặc cam, trong khi xoài sống (chua) có vỏ màu xanh lá. Dựa trên đặc điểm này, báo cáo tập trung xây dựng một phương pháp phân loại xoài chua/ngọt dựa vào màu sắc vỏ xoài, sử dụng kỹ thuật threshold-based classification.

Mục tiêu của bài toán là:

- Xây dựng hệ thống phân loại xoài chua/ngọt từ ảnh RGB
- Sử dụng đặc trưng màu đơn giản, dễ tính toán
- Áp dụng Leave-One-Out Cross Validation (LOOCV) để đánh giá mô hình một cách khách quan

1.2. Ý tưởng và đặc trưng sử dụng

Ý tưởng chính của phương pháp dựa trên nhận xét trực quan rằng:

- Xoài chua có tỷ lệ màu xanh lá (green channel) cao
- Xoài ngọt có tỷ lệ màu xanh lá thấp, thay vào đó là màu vàng/cam

Từ đó, một đặc trưng màu được đề xuất là tỷ lệ màu xanh lá trên tổng cường độ RGB, được tính theo công thức:

$$ratio\_green = \frac{\sum G}{\sum R + \sum G + \sum B}$$

H1. Công thức tính tỉ lệ màu xanh lá cây có trong ảnh

Trong đó:

- $\sum R, \sum G, \sum B$ lần lượt là tổng giá trị của các kênh màu đỏ, xanh lá và xanh dương
- lần lượt là tổng giá trị của các kênh màu đỏ, xanh lá và xanh dương
- Phép tính chỉ được thực hiện trên

1.3. Phương pháp phân loại

Bài toán được giải quyết bằng phương pháp phân loại dựa trên ngưỡng (threshold-based classification).
Ở mỗi lần huấn luyện, một ngưỡng được xác định từ dữ liệu huấn luyện, sau đó so sánh giá trị ratio_green của mẫu kiểm tra với ngưỡng này để đưa ra quyết định phân loại.

Phương pháp này có ưu điểm:

- Đơn giản, dễ triển khai
- Dễ giải thích và phù hợp cho mục đích học tập
- Hiệu quả với tập dữ liệu nhỏ

1.4. Giới thiệu công cụ

Các công cụ và thư viện được sử dụng trong báo cáo bao gồm:

Công cụ	Mục đích
Python 3.x	Ngôn ngữ lập trình
OpenCV (cv2)	Xử lý ảnh, tách nền
NumPy	Tính toán số học và xử lý mảng
Matplotlib	Trực quan hóa kết quả
Scikit-learn	Đánh giá mô hình (accuracy)

2. Phương pháp thực hiện

2.1. Tiền xử lý

a) Resize

- Tất cả các ảnh trong tập dữ liệu được chuẩn hóa về cùng kích thước 300 × 300 pixels.
- Việc resize ảnh giúp đảm bảo sự đồng nhất về kích thước đầu vào, thuận lợi cho quá trình xử lý ảnh và trích xuất đặc trưng, đồng thời giảm sự khác biệt do độ phân giải ảnh ban đầu gây ra.

b) Background remove

- Do ảnh có nền trắng, phương pháp tách nền dựa trên độ sáng không mang lại hiệu quả cao.Vì vậy, nền ảnh được loại bỏ bằng cách ngưỡng hóa màu trắng trong không gian RGB.

- Các điểm ảnh có giá trị R, G, B lớn hơn một ngưỡng xác định được xem là nền và bị loại bỏ. Sau đó, mặt nạ (mask) được đảo ngược để giữ lại vùng quả xoài. Các phép toán hình thái học được áp dụng nhằm loại bỏ nhiễu và làm mịn biên của vùng đối tượng.

```
IMG_SIZE = 300

def remove_background_grabcut(img):
    # Tạo mask
    mask = np.zeros(img.shape[:2], np.uint8)

    # Tạo background và foreground model
    bgdModel = np.zeros((1, 65), np.float64)
    fgdModel = np.zeros((1, 65), np.float64)

    # Định nghĩa rectangle bao quanh đối tượng (giả sử xoài ở giữa ảnh)
    height, width = img.shape[:2]
    rect = (int(width*0.1), int(height*0.1), int(width*0.8), int(height*0.8))

    # Áp dụng GrabCut
    cv2.grabCut(img, mask, rect, bgdModel, fgdModel, 5, cv2.GC_INIT_WITH_RECT)

    # Tạo mask nhị phân (0,2 = background, 1,3 = foreground)
    mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')

    # Làm sạch mask
    kernel = np.ones((7,7), np.uint8)
    mask2 = cv2.morphologyEx(mask2, cv2.MORPH_CLOSE, kernel)
    mask2 = cv2.morphologyEx(mask2, cv2.MORPH_OPEN, kernel)

    # Áp dụng mask lên ảnh gốc
    result = img * mask2[:, :, np.newaxis]

    return result, mask2 * 255

def preprocess_and_mask(img):
    img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
    mango_only, mask = remove_background_grabcut(img)
    return mango_only, mask
```

H2. Code resize + tách nền cho ảnh

c) Gán nhãn

Việc gán nhãn được thực hiện thủ công dựa trên màu sắc vỏ xoài:

- Xoài chua (sour): vỏ có màu xanh lá chiếm ưu thế
- Xoài ngọt (sweet): vỏ có màu vàng hoặc cam chiếm ưu thế

Dữ liệu được tổ chức thành hai thư mục:

- sweet → nhãn 0 (ngọt)
- sour → nhãn 1 (chua)

Cách gán nhãn này phù hợp với bài toán và không cần sử dụng thêm tệp nhãn riêng biệt.

```
# === GÁN NHÃN ===
# Sweet = 0
sweet_files = [f for f in os.listdir(SWEET_DIR) if
f.lower().endswith((".jpg",".png",".jpeg"))]
sweet_data = [(os.path.join(SWEET_DIR, f), 0) for f in sweet_files]
# Sour = 1
sour_files = [f for f in os.listdir(SOUR_DIR) if
f.lower().endswith((".jpg",".png",".jpeg"))]
sour_data = [(os.path.join(SOUR_DIR, f), 1) for f in sour_files]

labeled_data = sweet_data + sour_data
```

H3. Code gán nhãn

d) Kết quả

Sau bước tiền xử lý, tập dữ liệu thu được có các đặc điểm:

- Ảnh đã được resize về kích thước 300 × 300 pixels
- Nền trắng đã được loại bỏ hoàn toàn
- Chỉ giữ lại vùng quả xoài, trong khi phần nền được chuyển thành màu đen
- Màu sắc của quả xoài được bảo toàn, sẵn sàng cho bước trích xuất đặc trưng

H4. Kết quả sau khi tách nền

H5. Kết quả sau khi gán nhãn

2.2. Phân chia dữ liệu

Trong báo cáo này, dữ liệu không được chia theo cách truyền thống (train/test cố định), mà được đánh giá bằng phương pháp Leave-One-Out Cross Validation (LOOCV) nhằm đảm bảo tính khách quan với tập dữ liệu có kích thước nhỏ.

Cụ thể:

- Ở mỗi lần lặp, một ảnh xoài được chọn làm mẫu kiểm tra (test)
- Các ảnh còn lại được sử dụng làm tập huấn luyện (train)
- Quá trình được lặp lại cho đến khi mỗi ảnh đều được dùng làm mẫu kiểm tra đúng một lần

Với tập dữ liệu gồm N ảnh, phương pháp LOOCV thực hiện tổng cộng N lần huấn luyện và kiểm tra.Trong nghiên cứu này, với 22 ảnh xoài, mô hình được huấn luyện và đánh giá qua 22 vòng lặp.

Phương pháp LOOCV giúp tận dụng tối đa dữ liệu huấn luyện ở mỗi vòng lặp, đồng thời giảm sự phụ thuộc vào cách chia dữ liệu ngẫu nhiên, từ đó phản ánh chính xác hơn khả năng tổng quát hóa của mô hình.

```
# === LOOCV - PHÂN CHIA DỮ LIỆU ===
y_true = []
y_pred = []
thresholds_used = []

print("LOOCV - Phân chia dữ liệu:")
print("-" * 50)
```

```python
# === LOOCV - PHÂN CHIA DỮ LIỆU ===
n_samples = len(data)

print(f"PHÂN CHIA DỮ LIỆU - LOOCV:")
print(f"   - Tổng số mẫu: {n_samples}")
print(f"   - Mỗi iteration: Train = {n_samples-1} mẫu, Test = 1 mẫu")
print(f"   - Số lần lặp: {n_samples} iterations")
print("-" * 50)

# Tạo danh sách các fold cho LOOCV
loocv_folds = []
for i in range(n_samples):
    # Test: 1 mẫu (index i)
    test_idx = i
    # Train: tất cả mẫu còn lại
    train_idx = [j for j in range(n_samples) if j != i]
    loocv_folds.append((train_idx, test_idx))

    if i < 3:
        print(f"Fold {i+1}: Train indices = {train_idx[:3]}...{train_idx[-3:]} |
Test index = {test_idx}")

print("...")
print(f"\nĐã tạo {len(loocv_folds)} folds cho LOOCV")
```

H6. Code phân chia dữ liệu

```
PHÂN CHIA DỮ LIỆU - LOOCV:
   - Tổng số mẫu: 22
   - Mỗi iteration: Train = 21 mẫu, Test = 1 mẫu
   - Số lần lặp: 22 iterations
--------------------------------------------------
Fold 1: Train indices = [1, 2, 3]...[19, 20, 21] | Test index = 0
Fold 2: Train indices = [0, 2, 3]...[19, 20, 21] | Test index = 1
Fold 3: Train indices = [0, 1, 3]...[19, 20, 21] | Test index = 2
...

✅ Đã tạo 22 folds cho LOOCV
```

H7. Kết quả phân chia dữ liệu

2.3. Trích đặc trưng

Sau khi thực hiện tiền xử lý và tách nền, bước tiếp theo là trích xuất đặc trưng từ ảnh quả xoài để phục vụ cho quá trình phân loại. Do bài toán tập trung vào việc phân biệt xoài chua

và xoài ngọt dựa trên màu sắc vỏ, đặc trưng màu được lựa chọn là tỷ lệ màu xanh lá (ratio_green). Đặc trưng ratio_green được tính theo công thức:

$$ratio\_green = \frac{\sum G}{\sum R + \sum G + \sum B}$$

H8. Công thức tính tỉ lệ màu xanh lá cây có trong ảnh

Trong đó:

- $\sum R, \sum G, \sum B$lần lượt là tổng giá trị của các kênh màu đỏ, xanh lá và xanh dương
- Phép tính chỉ được thực hiện trên vùng quả xoài sau khi tách nền, nhằm tránh ảnh hưởng của nền ảnh đến kết quả

Đặc trưng ratio_green phản ánh mức độ "xanh" của vỏ xoài:

- Giá trị lớn cho thấy kênh xanh lá chiếm ưu thế, tương ứng với xoài còn xanh (chua)
- Giá trị nhỏ cho thấy xoài đã chín, màu vàng/cam chiếm ưu thế (ngọt)

Việc sử dụng đặc trưng màu đơn giản giúp giảm độ phức tạp của mô hình, đồng thời phù hợp với phương pháp phân loại dựa trên ngưỡng được áp dụng trong bài toán.

```
def extract_ratio_green(img_path):
    img = cv2.imread(img_path)
    mango, mask = preprocess_and_mask(img)

    B, G, R = cv2.split(mango)

    sum_r = np.sum(R[mask > 0])
    sum_g = np.sum(G[mask > 0])
    sum_b = np.sum(B[mask > 0])

    ratio = sum_g / (sum_r + sum_g + sum_b + 1e-6)
    return ratio
```

H9. Code Công thức tính tỉ lệ màu xanh lá cây có trong ảnh

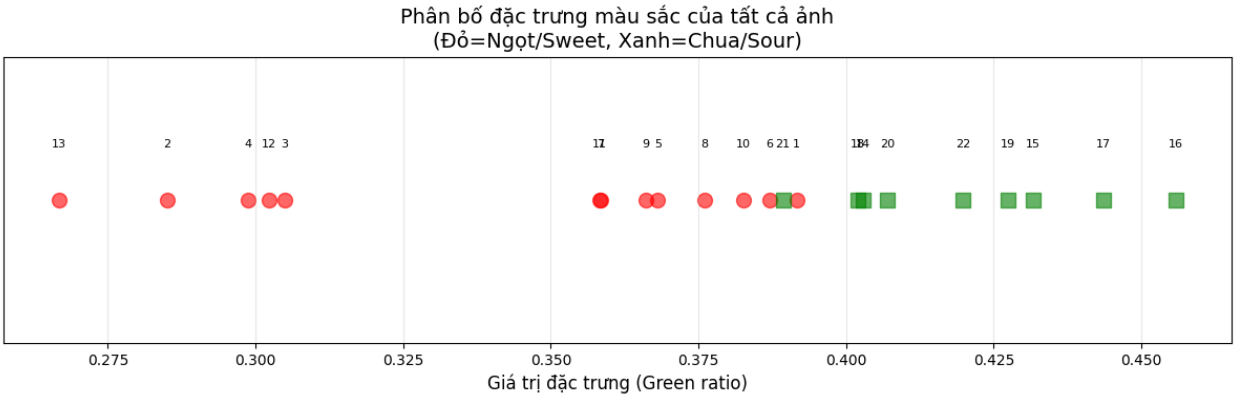
```
# === TÍNH ĐẶC TRƯNG CHO TẤT CẢ MẪU ===
data = []

for fpath, label in labeled_data:
```

```
    ratio = extract_ratio_green(fpath)
    data.append([ratio, label])

data = np.array(data, dtype=float)
```

H9. Code tính đặc trưng cho tất cả mẫu



H10. Kết quả đồ thị tính đặc trưng cho tất cả mẫu

2.4. Huấn luyện mô hình

Trong bài toán này, mô hình phân loại được huấn luyện bằng phương pháp phân loại dựa trên ngưỡng (threshold-based classification). Khác với các mô hình học máy phức tạp, phương pháp này không yêu cầu quá trình học tham số nhiều chiều mà chỉ cần xác định một giá trị ngưỡng trên đặc trưng đã trích xuất. Ở mỗi vòng lặp của phương pháp Leave-One-Out Cross Validation (LOOCV), quá trình huấn luyện được thực hiện như sau:

- Tập huấn luyện bao gồm tất cả các mẫu trừ đi một mẫu kiểm tra
- Từ tập huấn luyện, giá trị ngưỡng (threshold) được tính bằng trung bình của đặc trưng ratio_green
- Giá trị ngưỡng này đại diện cho ranh giới phân tách giữa hai lớp xoài chua và xoài ngọt

Sau khi xác định ngưỡng, mẫu kiểm tra được đưa vào mô hình và so sánh với ngưỡng để đưa ra kết quả phân loại. Phương pháp huấn luyện này có ưu điểm:

- Đơn giản, dễ cài đặt
- Không yêu cầu tập dữ liệu lớn
- Phù hợp với bài toán phân loại dựa trên đặc trưng màu

Công thức tính ngưỡng:

$$T = \frac{1}{N-1} \sum_{i=1}^{N-1} ratio\_green^{(i)}$$

H11. Công thức tính ngưỡng

Trong đó:

- $ratio\_green^{(i)}$ là giá trị đặc trưng ratio_green của mẫu thứ i trong tập huấn luyện
- N là tổng số mẫu trong tập dữ liệu

Sau khi xác định ngưỡng T, mẫu kiểm tra được phân loại theo quy tắc:

Nếu $ratio\_green > T \Rightarrow$ Xoài chua
Ngược lại $\Rightarrow$ Xoài ngọt

Quá trình trên được lặp lại cho từng mẫu trong tập dữ liệu theo phương pháp LOOCV, đảm bảo mỗi mẫu đều được sử dụng làm mẫu kiểm tra đúng một lần.

Phương pháp huấn luyện này đơn giản, dễ triển khai và phù hợp với bài toán phân loại dựa trên màu sắc của quả xoài.

```
# === LOOCV - TRAIN MODEL ===
y_true = []
y_pred = []
thresholds_used = []

print("BẮT ĐẦU TRAIN MODEL VỚI LOOCV:")
print("-" * 60)

for fold_idx, (train_idx, test_idx) in enumerate(loocv_folds):
    # Lấy dữ liệu train và test
    train_data = data[train_idx]
    test_ratio = data[test_idx, 0]
    test_label = int(data[test_idx, 1])

    # TRAIN: Tính ngưỡng từ tập train (mean của tất cả ratio)
    threshold = np.mean(train_data[:, 0])
    thresholds_used.append(threshold)
```

```
    # TEST: Dự đoán nhãn
    # Nếu ratio >= threshold → Sour (1), ngược lại → Sweet (0)
    pred_label = 1 if test_ratio >= threshold else 0

    y_true.append(test_label)
    y_pred.append(pred_label)

    if fold_idx < 5:
        status = "✓" if test_label == pred_label else "✗"
        print(f"Fold {fold_idx+1:2d}: Threshold={threshold:.4f} | Test
ratio={test_ratio:.4f} | True={test_label} Pred={pred_label} {status}")

print("...")
print(f"\n Hoàn thành train {len(loocv_folds)} folds")
print(f"   - Đã tạo {len(thresholds_used)} threshold")
print(f"   - Threshold trung bình: {np.mean(thresholds_used):.4f}")
```

H12. Code train model

```
BẮT ĐẦU TRAIN MODEL VỚI LOOCV:
------------------------------------------------------------
Fold  1: Threshold=0.3716 | Test ratio=0.3917 | True=0 Pred=1 ✗
Fold  2: Threshold=0.3767 | Test ratio=0.2850 | True=0 Pred=0 ✓
Fold  3: Threshold=0.3758 | Test ratio=0.3049 | True=0 Pred=0 ✓
Fold  4: Threshold=0.3761 | Test ratio=0.2988 | True=0 Pred=0 ✓
Fold  5: Threshold=0.3728 | Test ratio=0.3681 | True=0 Pred=0 ✓
...

✅ Hoàn thành train 22 folds
   - Đã tạo 22 threshold
   - Threshold trung bình: 0.3726
```

H13. Kết quả train model

2.5. Đánh giá mô hình

2.5.1. Chỉ số đánh giá

Mô hình được đánh giá bằng phương pháp Leave-One-Out Cross Validation (LOOCV). Sau khi hoàn thành toàn bộ các vòng lặp LOOCV, độ chính xác được tính bằng:

$$Accuracy = \frac{\text{số mẫu được phân loại đúng}}{\text{tổng số mẫu}}$$

H14. Công thức tính độ chính xác

Trong đó:

- Số mẫu được phân loại đúng là tổng số mẫu mà nhãn dự đoán trùng với nhãn thực
- Tổng số mẫu là tổng số ảnh trong tập dữ liệu

```
acc_loocv = accuracy_score(y_true, y_pred)
cm_loocv = confusion_matrix(y_true, y_pred, labels=[0, 1])
```

H15. Code công thức tính độ chính xác

Trong mỗi vòng LOOCV, một giá trị ngưỡng (threshold) được tính từ tập huấn luyện. Sau khi hoàn tất LOOCV, giá trị ngưỡng trung bình được xác định:

$$T_{mean} = \frac{1}{N}\sum_{i=1}^{N} T_i$$

H16. Công thức tính ngưỡng trung bình

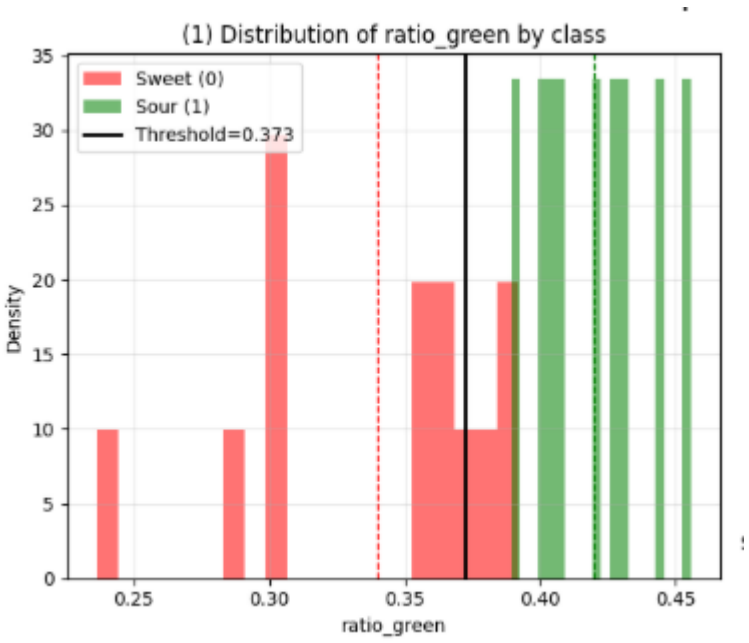
```
threshold_mean = np.mean(thresholds_used)
```

H17. Code công thức tính ngưỡng trung bình

Giá trị này đại diện cho ranh giới phân tách trung bình giữa hai lớp xoài chua và xoài ngọt trong toàn bộ quá trình đánh giá.

2.5.2. Phân tích các biểu đồ trực quan

2.5.2.1. Histogram phân bố ratio_green theo lớp



H18. Đồ thị phân bố ratio_green theo lớp

Biểu đồ histogram thể hiện phân bố đặc trưng ratio_green của hai lớp:

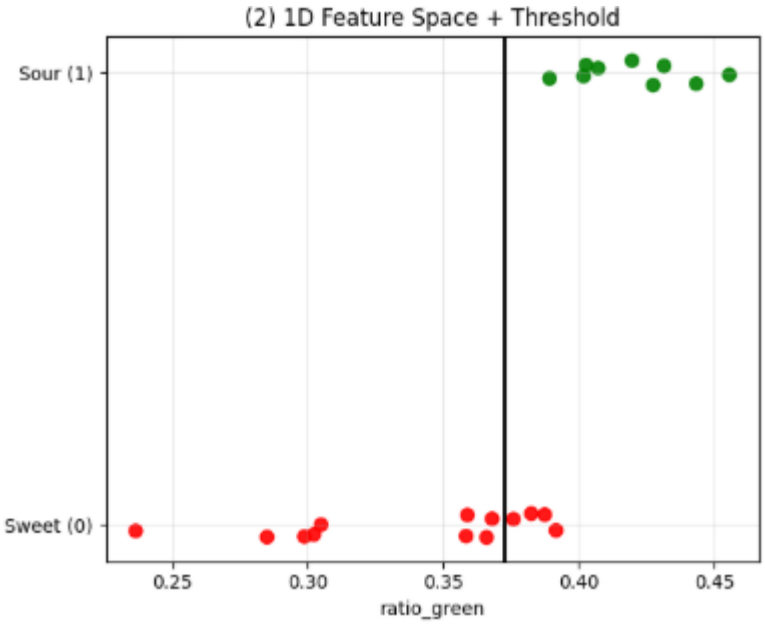
- Sweet (0) – màu đỏ
- Sour (1) – màu xanh

Các đường nét đứt biểu diễn giá trị trung bình ratio_green của từng lớp, trong khi đường thẳng màu đen biểu diễn ngưỡng trung bình.

Quan sát cho thấy:

- Xoài chua có xu hướng ratio_green cao hơn
- Xoài ngọt có ratio_green thấp hơn
- Hai phân bố có sự tách biệt tương đối rõ ràng

2.5.2.2. Không gian đặc trưng 1 chiều + ngưỡng



H19. Đồ thị không gian đặc trưng 1 chiều + ngưỡng

Biểu đồ scatter thể hiện từng mẫu dữ liệu trong không gian đặc trưng một chiều (ratio_green).

Trục tung chỉ mang tính phân biệt lớp (Sweet/Sour), trong khi trục hoành là giá trị đặc trưng.

Đường thẳng đứng biểu diễn ngưỡng phân loại:

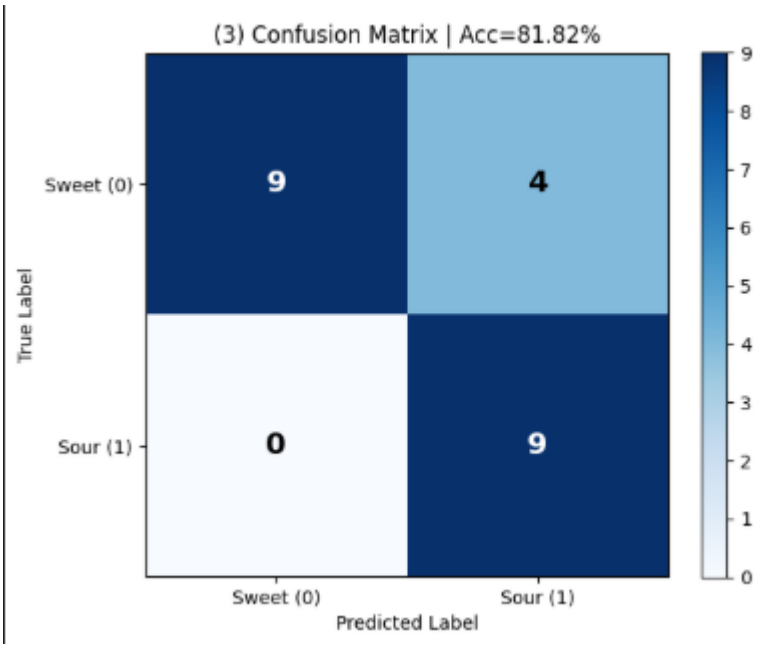
- Các điểm nằm bên phải ngưỡng → dự đoán xoài chua
- Các điểm nằm bên trái ngưỡng → dự đoán xoài ngọt

Biểu đồ này minh họa trực quan cách mô hình đưa ra quyết định phân loại.

2.5.2.3. Confusion Matrix

Confusion matrix cho biết số lượng dự đoán đúng và sai của mô hình:

- True Negative (TN): xoài ngọt được phân loại đúng là ngọt
- False Positive (FP): xoài ngọt bị phân loại nhầm thành chua
- False Negative (FN): xoài chua bị phân loại nhầm thành ngọt
- True Positive (TP): xoài chua được phân loại đúng là chua

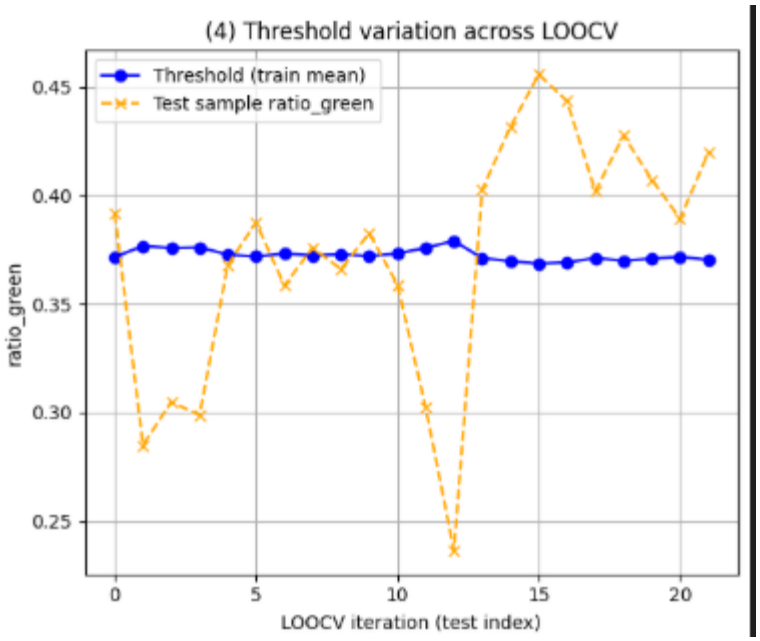


H20. Confusion Matrix

```
Confusion Matrix:
- True Negative (Sweet đúng): 9
- False Positive (Sweet → Sour): 4
- False Negative (Sour → Sweet): 0
- True Positive (Sour đúng): 9
```

H21. Kết quả confusion matrix

2.5.2.4. Biến thiên ngưỡng qua các vòng LOOCV



H22. Biến thiên ngưỡng qua các vòng LOOCV

Biểu đồ này thể hiện:

- Đường màu xanh: giá trị ngưỡng được tính ở mỗi vòng LOOCV
- Đường màu cam: giá trị ratio_green của mẫu test ở vòng tương ứng

Biểu đồ cho thấy:

- Ngưỡng thay đổi nhẹ giữa các vòng lặp
- Giá trị ratio_green của mẫu test có thể nằm hai phía của ngưỡng
- Mô hình nhạy cảm với một số mẫu biên, phản ánh tính chất của tập dữ liệu nhỏ

2.6. Kết quả

Sau khi thực hiện đánh giá mô hình bằng phương pháp Leave-One-Out Cross Validation (LOOCV) trên tập dữ liệu gồm 22 ảnh xoài, mô hình đạt được các kết quả sau:

- Độ chính xác: 81.82% (18/22 mẫu được phân loại đúng)
- Giá trị ngưỡng trung bình (Average threshold): 0.3726

Kết quả cho thấy phương pháp phân loại dựa trên ngưỡng với đặc trưng ratio_green có khả năng phân biệt xoài chua và xoài ngọt tương đối tốt.Mặc dù mô hình có cấu trúc đơn giản, độ chính xác đạt được là khả quan đối với tập dữ liệu nhỏ.

Nhận xét:

- Phần lớn các mẫu được phân loại chính xác, chỉ có 4/22 mẫu bị phân loại sai
- Các mẫu bị nhầm lẫn chủ yếu có giá trị ratio_green nằm gần ngưỡng phân loại

- Giá trị ngưỡng trung bình ổn định, phản ánh sự nhất quán của mô hình qua các vòng LOOCV

3. Kết luận

- Trong bài tập này, nhóm đã xây dựng quy trình phân loại độ ngọt của quả xoài dựa trên xử lý ảnh và phương pháp phân ngưỡng. Hệ thống đã thực hiện đầy đủ các bước từ tiền xử lý (tách nền), trích xuất đặc trưng ratio_green đến huấn luyện và kiểm thử mô hình theo phương pháp LOOCV trên tập dữ liệu 22 ảnh .
  Kết quả thực nghiệm cho thấy:
  - Đặc trưng màu sắc (tỷ lệ kênh Green trên tổng thể RGB) là một chỉ số hiệu quả để phân biệt giữa xoài chua (vỏ xanh) và xoài ngọt (vỏ chín vàng/đỏ).
  - Mô hình đạt độ chính xác là 18/22 (tương đương 81,82 %).
  - Ngưỡng phân loại (Threshold) được tính toán động dựa trên trung bình cộng của tập huấn luyện giúp mô hình thích nghi tốt hơn với sự biến thiên của dữ liệu.
- Ưu điểm và hạn chế của phương pháp
  Ưu điểm: Chi phí tính toán thấp: Thuật toán trích xuất đặc trưng và so sánh ngưỡng rất đơn giản, tốc độ xử lý nhanh, có thể ứng dụng trên các thiết bị cấu hình thấp.
  Hạn chế:
  - Phụ thuộc vào điều kiện ánh sáng: Do sử dụng không gian màu RGB, giá trị đặc trưng ratio_green dễ bị ảnh hưởng khi cường độ sáng thay đổi (ví dụ: ảnh quá tối hoặc bị lóa sáng), dẫn đến việc tính toán tỷ lệ màu không chính xác.
  - Phụ thuộc vào bước tiền xử lý: Độ chính xác của bài toán phụ thuộc rất lớn vào bước tách nền . Nếu tách nền không sạch, giá trị đặc trưng sẽ bị sai lệch.

- Trong tương lai, có thể cải thiện kết quả bằng cách loại bỏ hoàn toàn phần lá xoài, kết hợp thêm các đặc trưng màu khác hoặc áp dụng các mô hình học máy nâng cao hơn trên tập dữ liệu lớn hơn.