ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN TỬ - VIỄN THÔNG

TRÍ TUỆ NHÂN TẠO
BÁO CÁO BÀI TẬP LỚN 2

ĐỀ TÀI: XÂY DỰNG HỆ THỐNG PHÂN LOẠI HOA DIÊN VĨ, DÙNG CƠ SỞ DỮ LIỆU IRIS FLOWER

Sinh viên thực hiện:	Đinh Văn Quang	20DT1
	Hà Phước Phúc	20DT2
	Nguyễn Văn Qúy	21DT2
Lớp học phần:	20.38	
Giảng viên hướng dẫn:	TS. Hoàng Lê Uyên Thục	

Đà Nẵng, 10/2024

1. Giới thiệu

1.1 Tính cấp thiết

Hệ thống phân loại hoa diên vĩ sử dụng phương pháp KNN là một ứng dụng thực tiễn có thể giúp nâng cao nhận thức về các loài hoa, hỗ trợ nông dân trong việc chọn giống, và phục vụ cho mục đích giáo dục. Việc sử dụng cơ sở dữ liệu hoa diên vĩ cho phép triển khai nhanh chóng và hiệu quả.

Bài toán phân loại hoa diên vĩ là một ví dụ cơ bản giúp hiểu các thuật toán học máy trong phân loại dữ liệu. Việc giải quyết bài toán này rèn luyện kỹ năng tiền xử lý dữ liệu, chọn mô hình, và đánh giá hiệu quả mô hình. Ứng dụng của nó trải rộng trong nhiều lĩnh vực như y tế, nông nghiệp, và tự động hóa, giúp giảm thời gian, nguồn lực. Bài toán còn là bước khởi đầu cho các bài toán phức tạp hơn.

1.2 Giới thiệu bài toán

- Trong bài toán này, chúng ta cần xây dựng một hệ thống phân loại để xác định loài của hoa diên vĩ (Iris) dựa trên các đặc điểm của nó, chẳng hạn như chiều dài và chiều rộng của đài hoa và cánh hoa. Dữ liệu sử dụng là bộ cơ sở dữ liệu Iris flower dataset, được thu thập và công bố lần đầu bởi nhà thống kê học Ronald A. Fisher vào năm 1936.
- Cơ sở dữ liệu hoa diên vĩ gồm 150 mẫu hoa thuộc ba loài khác nhau của hoa diên vĩ: Iris-setosa, Iris-versicolor và Iris-virginica.
- Mỗi mẫu được mô tả qua 4 đặc trưng: Chiều dài đài hoa, chiều rộng đài hoa, chiều dài cánh hoa, chiều rộng cánh hoa.
- Nhiệm vụ của hệ thống là dựa vào các đặc trưng trên để phân loại đúng loài hoa. Trong bài toán này, chúng ta sẽ thử nghiệm với hai giá trị K khác nhau là K = 3 và
K = 5 để xem ảnh hưởng của việc lựa chọn K đến hiệu quả của hệ thống phân loại.
- Để đánh giá hiệu quả của hệ thống phân loại, chúng ta sẽ sử dụng ba phương pháp đánh giá phổ biến:
 + ***Phương pháp 1:*** Hold-out Cross Validation (Tỉ lệ 30:70): Dữ liệu được chia thành hai phần, trong đó 70% được dùng làm dữ liệu huấn luyện và 30% làm dữ liệu kiểm tra.
 + ***Phương pháp 2:*** K-Fold Cross Validation (K=5): Trong phương pháp này, dữ liệu sẽ được chia thành 5 phần (folds). Ở mỗi lần chạy, 4 phần sẽ được sử dụng để huấn luyện và 1 phần còn lại để kiểm tra. Quá trình này được lặp lại 5 lần, đảm bảo mỗi phần dữ liệu đều được kiểm tra một lần.
 + ***Phương pháp 3:*** Leave-One-Out Cross Validation (LOOCV): Mỗi mẫu dữ liệu sẽ được lần lượt chọn làm dữ liệu kiểm tra, trong khi các mẫu còn lại sẽ dùng để huấn luyện.
- Mục tiêu của hệ thống là:
 + Xây dựng hệ thống phân loại hoa diên vĩ dựa trên thuật toán KNN.
 + Thử nghiệm với các giá trị K = 3 và K = 5.
 + Đánh giá hiệu quả phân loại bằng ba phương pháp đánh giá khác nhau.

1.3 Giới thiệu phương pháp thực hiện

- Phương pháp K-nearest neighbor (KNN) là một trong những thuật toán học có giám sát đơn giản nhất trong học máy. Khi huấn luyện, thuật toán này không học một điều gì từ dữ liệu huấn luyện mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới.

K-nearest neighbor có thể áp dụng được vào cả hai loại bài toán học có giám sát là phân loại và hồi quy.

- Phương pháp KNN hoạt động dựa trên nguyên tắc rằng các mẫu tương tự sẽ nằm gần nhau trong không gian đặc trưng. Khi một điểm dữ liệu mới được đưa vào, KNN sẽ tìm K điểm gần nhất từ tập huấn luyện và sử dụng thông tin của chúng để xác định phân loại hoặc giá trị cho điểm dữ liệu đó.

Bước 1: Chọn số K: Cần chọn số lượng điểm lân cận K để xem xét. Thông thường, K là một số nguyên dương và là số lẻ để tránh trường hợp tranh chấp trong phân loại.

Bước 2: Tính khoảng cách: Tính khoảng cách giữa điểm dữ liệu mới và tất cả các điểm trong tập huấn luyện, khoảng cách có thể được tính theo Euclidean, Manhattan.

- Công thức Euclidean

$$\sqrt{\sum_{i=1}^{n}|x_i - y_i|^2}$$

- Công thức Manhattan

$$\sum_{i=1}^{n}|x_i - y_i|$$

Bước 3: Chọn K điểm gần nhất: Sau khi tính khoảng cách, chọn K điểm dữ liệu gần nhất với điểm cần phân loại.

Bước 4: Phân loại hoặc hồi quy:

- Phân loại: Nếu bài toán là phân loại, mô hình sẽ xác định lớp của điểm dữ liệu mới bằng cách lấy lớp chiếm ưu thế trong K điểm gần nhất.
- Hồi quy: Nếu bài toán là hồi quy, mô hình sẽ tính giá trị trung bình (hoặc trung vị) của các giá trị của K điểm gần nhất để đưa ra dự đoán.

2. Thực hiện

2.1 Các bước thực hiện

Bước 1. Mô tả dữ liệu

- Bộ dữ liệu 150 mẫu hoa diên vĩ gồm 3 lớp, mỗi lớp có 50 mẫu, trong đó mỗi lớp đề cập đến một loại cây hoa diên vĩ (Iris Setosa, Iris Versicolor và Iris Virginica).
- Vector đặc trưng 4 chiều (4D) gồm chiều dài cánh hoa, chiều rộng cánh hoa, chiều dài lá đài, chiều rộng lá đài.
- Dữ liệu tập train gồm 147/150 mẫu
- Dữ liệu tập test gồm 3 mẫu được rút ra từ cơ sở dữ liệu ban đầu:
  + Mẫu 1: 5.1, 3.5, 1.4, 0.2, Iris-setosa
  + Mẫu 51: 7.0, 3.2, 4.7, 1.4, Iris-versicolor
  + Mẫu 101: 6.3,3.3,6.0,2.5, Iris-virginica

Bước 2. Tiền xử lý dữ liệu

- Dữ liệu 150 mẫu hoa đã được xử lý, gán nhãn và tính đặc trưng

Bước 3. Rút trích đặc trưng

- Mỗi mẫu hoa là một vector 4D
- Vector gồm 4 thông số: Chiều dài cánh hoa, chiều rộng cánh hoa, chiều dài lá đài, chiều rộng lá đài

Bước 4. Xác định mô hình

- Tính khoảng cách từ vector test đến tất cả các vector train theo khoảng cách Mahattan (Norm 1) và Euclidean (Norm 2)

```
def manhattan_distance(row):  1 usage
    return np.sum(np.abs(test_vector - row.iloc[:4]))
def euclidean_distance(row):  1 usage
    return round(np.sqrt(np.sum(np.square(test_vector - row.iloc[:4]))), 2)

# Tính khoảng cách cho từng mẫu và thêm vào cột '_Distance'
data_iris['Manhattan_Distance'] = data_iris.apply(manhattan_distance, axis=1)
data_iris['Euclidean_Distance'] = data_iris.apply(euclidean_distance, axis=1)
```

Bước 5. Phân loại và kết luận (K=3 và K=5)

- Sắp xếp và chọn ra K khoảng cách nhỏ nhất/ngắn nhất trong 147 khoảng cách theo Mahattan và Euclidean
- Xem trong K khoảng cách nhỏ nhất đó thuộc lớp (class) nào chiếm ưu thế hơn thì kết luận mẫu test thuộc về lớp đó

```
# Sắp xếp theo khoảng cách và lấy K hàng đầu tiên
nearest_manhattan = data_iris.nsmallest(k_value, columns: 'Manhattan_Distance')
nearest_euclidean = data_iris.nsmallest(k_value, columns: 'Euclidean_Distance')

print(f"\n{k_value} mẫu có khoảng cách Manhattan nhỏ nhất:")
print(nearest_manhattan[['Class', 'Manhattan_Distance']])

print(f"\n{k_value} mẫu có khoảng cách Euclidean nhỏ nhất:")
print(nearest_euclidean[['Class', 'Euclidean_Distance']])

# Xác định lớp chiếm ưu thế cho khoảng cách
manhattan_classes = nearest_manhattan['Class'].tolist()
manhattan_most_common = Counter(manhattan_classes).most_common(1)[0]
print(f"\n(Mahattan distance) Vector test thuộc Class: "
      f"{manhattan_most_common[0]} ({manhattan_most_common[1]} lần)")

euclidean_classes = nearest_euclidean['Class'].tolist()
euclidean_most_common = Counter(euclidean_classes).most_common(1)[0]
print(f"(Euclidean distance) Vector test thuộc Class: "
      f"{euclidean_most_common[0]} ({euclidean_most_common[1]} lần)")
```

Bước 6. Đánh giá độ chính xác của hệ thống

a. Phương pháp Hold-out

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier

# Định nghĩa tên cột
columns = ['sepal length', 'sepal width', 'petal length', 'petal width', 'class']

# Tải tập dữ liệu
data = pd.read_csv("Program/python/iris/iris.data", header=None, names=columns)

print(data)

# Chuẩn bị dữ liệu
X = data.iloc[:, 0:4].values
y = data['class'].values


# Hàm tính độ chính xác
def accuracy_score(y_true, y_pred):
    return np.mean(y_true == y_pred)
```

```python
# Phương pháp Hold-out
def holdout_method(X, y, test_size=0.3, random_state=42):
    np.random.seed(random_state)

    # Tạo dictionary để lưu trữ indices cho mỗi class
    class_indices = {cls: np.where(y == cls)[0] for cls in np.unique(y)}

    X_train, X_test, y_train, y_test = [], [], [], []

    for cls, indices in class_indices.items():
        np.random.shuffle(indices)
        split_point = int(len(indices) * (1 - test_size))

        X_train.extend(X[indices[:split_point]])
        X_test.extend(X[indices[split_point:]])
        y_train.extend([cls] * split_point)
        y_test.extend([cls] * (len(indices) - split_point))

    X_train, X_test = np.array(X_train), np.array(X_test)
    y_train, y_test = np.array(y_train), np.array(y_test)

    model = KNeighborsClassifier(n_neighbors=5)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

# Tính độ chính xác cho từng class
class_accuracies = {}
for cls in np.unique(y):
    cls_indices = y_test == cls
    cls_correct = np.sum((y_test[cls_indices] == y_pred[cls_indices]))
    cls_total = np.sum(cls_indices)
    class_accuracies[cls] = cls_correct / cls_total

# Tính độ chính xác tổng thể
overall_accuracy = np.mean(y_test == y_pred) * 100

return overall_accuracy, class_accuracies, y_test, y_pred
```

b. Phương pháp K-fold với K=5

```
# Phương pháp K-fold Cross-validation
def k_fold_cross_validation(X, y, k=5):
    fold_size = len(X) // k
    indices = np.arange(len(X))
    np.random.shuffle(indices)

    scores = []
    for i in range(k):
        start = i * fold_size
        end = (i + 1) * fold_size if i < k - 1 else len(X)
        test_indices = indices[start:end]
        train_indices = np.concatenate([indices[:start], indices[end:]])

        X_train, X_test = X[train_indices], X[test_indices]
        y_train, y_test = y[train_indices], y[test_indices]

        model = KNeighborsClassifier(n_neighbors=5)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        accuracy = accuracy_score(y_test, y_pred)
        scores.append(accuracy)

    return np.mean(scores) * 100
```

Giải thích:

- K-fold cross-validation: Sử dụng K-fold từ thư viện sklearn để chia tập dữ liệu thành 5 folds. Trong mỗi vòng lặp, một fold được dùng để kiểm tra, 4 folds còn lại để huấn luyện.
- KNN classification: Mỗi lần lặp trong K-fold, tập huấn luyện và kiểm tra được truyền vào hàm knn_predict, nơi mà Manhattan distance được sử dụng để tính toán và dự đoán.
- Tính độ chính xác: Sau khi dự đoán, độ chính xác của từng fold được tính bằng accuracy_score, và sau đó tính trung bình qua tất cả 5 folds.

c. Phương pháp LOOCV

```
# Phương pháp Leave-One-Out Cross-validation (LOOCV)
def loocv(X, y):
    n_samples = len(X)
    correct_predictions = 0

    for i in range(n_samples):
        # Lấy 1 mẫu làm test, còn lại làm train
        X_test = X[i:i + 1]
        y_test = y[i]
        X_train = np.concatenate([X[:i], X[i + 1:]], axis=0)
        y_train = np.concatenate([y[:i], y[i + 1:]])

        # Huấn luyện mô hình trên tập train
        model = KNeighborsClassifier(n_neighbors=5)
        model.fit(X_train, y_train)

        # Dự đoán nhãn cho mẫu test
        y_pred = model.predict(X_test)

        # Kiểm tra dự đoán
        if y_pred == y_test:
            correct_predictions += 1

    # Tính độ chính xác
    accuracy = (correct_predictions / n_samples) * 100
    return accuracy


# Thực hiện các phương pháp đánh giá
overall_accuracy, class_accuracies, y_test, y_pred = holdout_method(X, y)
k_fold_accuracy = k_fold_cross_validation(X, y)
loocv_accuracy = loocv(X, y)
# In kết quả
print("\nHold-out Method:")
print(f"Độ chính xác tổng thể: {overall_accuracy:.2f}%")
print("Độ chính xác theo từng class:")
for cls, accuracy in class_accuracies.items():
    print(f"  {cls}: {accuracy:.2f}")

print("\nK-fold Cross-validation (k=5):")
print(f"Độ chính xác trung bình: {k_fold_accuracy:.2f}%")

print("\nLeave-One-Out Cross-validation (LOOCV):")
print(f"Độ chính xác: {loocv_accuracy:.2f}%")
```

2.2 Kết quả

a. Với K = 3:

- Mẫu test 1: *5.1,3.5,1.4,0.2* thuộc class *Iris-setosa*

```
Nhập giá trị K: 3
Nhập vector test gồm 4 đặc trưng: 5.1,3.5,1.4,0.2

3 mẫu có khoảng cách Manhattan nhỏ nhất:
          Class  Manhattan_Distance
16  Iris-setosa                 0.1
3   Iris-setosa                 0.2
38  Iris-setosa                 0.2

3 mẫu có khoảng cách Euclidean nhỏ nhất:
          Class  Euclidean_Distance
16  Iris-setosa                0.10
3   Iris-setosa                0.14
26  Iris-setosa                0.14

(Mahattan distance) Vector test thuộc Class: Iris-setosa (3 lần)
(Euclidean distance) Vector test thuộc Class: Iris-setosa (3 lần)
```

- Mẫu test 51: *7.0,3.2,4.7,1.4* thuộc class *Iris-versicolor*

```
Nhập giá trị K: 3
Nhập vector test gồm 4 đặc trưng: 7.0,3.2,4.7,1.4

3 mẫu có khoảng cách Manhattan nhỏ nhất:
              Class  Manhattan_Distance
50  Iris-versicolor                 0.5
84  Iris-versicolor                 0.5
63  Iris-versicolor                 0.7

3 mẫu có khoảng cách Euclidean nhỏ nhất:
              Class  Euclidean_Distance
50  Iris-versicolor                0.26
84  Iris-versicolor                0.33
63  Iris-versicolor                0.44

(Mahattan distance) Vector test thuộc Class: Iris-versicolor (3 lần)
(Euclidean distance) Vector test thuộc Class: Iris-versicolor (3 lần)
```

- Mẫu test 101: *6.3,3.3,6.0,2.5* thuộc class *Iris-virginica*

```
Nhập giá trị K: 3
Nhập vector test gồm 4 đặc trưng: 6.3,3.3,6.0,2.5

3 mẫu có khoảng cách Manhattan nhỏ nhất:
              Class  Manhattan_Distance
133  Iris-virginica                 0.6
141  Iris-virginica                 0.7
140  Iris-virginica                 0.9

3 mẫu có khoảng cách Euclidean nhỏ nhất:
              Class  Euclidean_Distance
133  Iris-virginica                0.42
141  Iris-virginica                0.50
101  Iris-virginica                0.51

(Mahattan distance) Vector test thuộc Class: Iris-virginica (3 lần)
(Euclidean distance) Vector test thuộc Class: Iris-virginica (3 lần)
```

b. Với K = 5:

- Mẫu test 1: *5.1,3.5,1.4,0.2* thuộc class *Iris-setosa*

```
Nhập giá trị K: 5
Nhập vector test gồm 4 đặc trưng: 5.1,3.5,1.4,0.2

5 mẫu có khoảng cách Manhattan nhỏ nhất:
          Class  Manhattan_Distance
16  Iris-setosa                 0.1
3   Iris-setosa                 0.2
38  Iris-setosa                 0.2
26  Iris-setosa                 0.2
27  Iris-setosa                 0.2

5 mẫu có khoảng cách Euclidean nhỏ nhất:
          Class  Euclidean_Distance
16  Iris-setosa                0.10
3   Iris-setosa                0.14
26  Iris-setosa                0.14
27  Iris-setosa                0.14
38  Iris-setosa                0.14

(Mahattan distance) Vector test thuộc Class: Iris-setosa (5 lần)
(Euclidean distance) Vector test thuộc Class: Iris-setosa (5 lần)
```

- Mẫu test 51: *7.0,3.2,4.7,1.4* thuộc class *Iris-versicolor*

```
Nhập giá trị K: 5
Nhập vector test gồm 4 đặc trưng: 7.0,3.2,4.7,1.4

5 mẫu có khoảng cách Manhattan nhỏ nhất:
              Class  Manhattan_Distance
50  Iris-versicolor                 0.5
84  Iris-versicolor                 0.5
63  Iris-versicolor                 0.7
74  Iris-versicolor                 0.7
49  Iris-versicolor                 0.9

5 mẫu có khoảng cách Euclidean nhỏ nhất:
              Class  Euclidean_Distance
50  Iris-versicolor                0.26
84  Iris-versicolor                0.33
63  Iris-versicolor                0.44
74  Iris-versicolor                0.46
56  Iris-versicolor                0.52

(Mahattan distance) Vector test thuộc Class: Iris-versicolor (5 lần)
(Euclidean distance) Vector test thuộc Class: Iris-versicolor (5 lần)
```

- Mẫu test 101: *6.3,3.3,6.0,2.5* thuộc class *Iris-virginica*

```
Nhập giá trị K: 5
Nhập vector test gồm 4 đặc trưng: 6.3,3.3,6.0,2.5

5 mẫu có khoảng cách Manhattan nhỏ nhất:
              Class  Manhattan_Distance
133  Iris-virginica                 0.6
141  Iris-virginica                 0.7
140  Iris-virginica                 0.9
145  Iris-virginica                 1.0
101  Iris-virginica                 1.0

5 mẫu có khoảng cách Euclidean nhỏ nhất:
              Class  Euclidean_Distance
133  Iris-virginica                0.42
141  Iris-virginica                0.50
101  Iris-virginica                0.51
140  Iris-virginica                0.56
137  Iris-virginica                0.61

(Mahattan distance) Vector test thuộc Class: Iris-virginica (5 lần)
(Euclidean distance) Vector test thuộc Class: Iris-virginica (5 lần)
```

3. Kết luận

3.1 Nhận xét

```
Hold-out Method:
Độ chính xác tổng thể: 97.78%
Độ chính xác theo từng class:
  Iris-setosa: 1.00
  Iris-versicolor: 1.00
  Iris-virginica: 0.93

K-fold Cross-validation (k=5):
Độ chính xác trung bình: 97.33%

Leave-One-Out Cross-validation (LOOCV):
Độ chính xác: 96.67%
```

a. Phương pháp 1: Đánh giá theo phương pháp Hold-out 30:70

- Chia dữ liệu với tỷ lệ 30:70 sẽ chia dữ liệu thành 70% cho huấn luyện và 30% còn lại để kiểm tra mô hình KNN.
- Độ chính xác của phương pháp Hold-out là 97.78%.
- Nhận xét:
  + Hiệu suất cao: Độ chính xác cao cho thấy rằng mô hình đã học được các đặc điểm quan trọng từ dữ liệu huấn luyện và có khả năng áp dụng các đặc điểm này để phân loại đúng các mẫu dữ liệu mới.

+ Mẫu dữ liệu tốt: Độ chính xác này cũng có thể chỉ ra rằng dữ liệu có chất lượng tốt, nghĩa là có sự phân chia rõ ràng giữa các lớp hoặc nhãn mà mô hình đang cố gắng phân loại.

b. Phương pháp 2: Đánh giá theo phương pháp K-fold (với K = 5)

- Để sử dụng k-fold cross-validation với K=5, bạn có thể áp dụng cách chia tập dữ liệu thành 5 phần (folds). Trong mỗi lần lặp, một phần sẽ được sử dụng làm tập kiểm tra và 4 phần còn lại làm tập huấn luyện. Sau đó, tính trung bình độ chính xác của mô hình qua 5 lần lặp.
- Phương pháp đánh giá K-fold với K=5 với độ chính xác trung bình của mô hình qua 5 lần lặp là 97.33%.
- Nhận xét:

+ Độ chính xác trung bình 97.33% cho thấy mô hình hoạt động rất tốt trên hầu hết các tập dữ liệu. Điều này cho thấy rằng mô hình có khả năng phân loại chính xác các mẫu mới.

c. Phương pháp 3: Đánh giá theo phương pháp LOOCV

- Độ chính xác của phương pháp LOOCV là 96.67%
- Nhận xét:

+ Độ chính xác khá cao, cho thấy mô hình có khả năng phân loại chính xác hầu hết các mẫu trong tập dữ liệu. Điều này cho thấy rằng mô hình đã học được các đặc điểm quan trọng để phân loại.

+ Dù độ chính xác cao là tốt, nhưng trong một số trường hợp, nó có thể cho thấy rằng mô hình đã bị overfit, nếu nhận thấy sự chênh lệch lớn giữa độ chính xác trên dữ liệu huấn luyện và độ chính xác trên dữ liệu kiểm tra.

3.2 Ưu và khuyết điểm

a. Ưu điểm

- Đơn giản và dễ giải thích
- Không dựa trên bất kỳ giả định nào, vì thế nó có thể được sử dụng trong các bài toán phi tuyến tính.
- Hoạt động tốt trong trường hợp phân loại với nhiều lớp
- Sử dụng được trong cả phân loại và hồi quy

b. Nhược điểm

- Trở nên rất chậm khi số lượng điểm dữ liệu tăng lên vì mô hình cần lưu trữ tất cả các điểm dữ liệu.
- Tốn bộ nhớ
- Nhạy cảm với các dữ liệu bất thường (nhiễu)

3.3 Hướng phát triển:

- Mở rộng dữ liệu: Sử dụng các kỹ thuật tăng cường dữ liệu để tạo ra nhiều mẫu hơn từ dữ liệu hiện có.
- Học bán giám sát: sử dụng cả dữ liệu có nhãn và không nhãn, cải thiện hiệu suất khi có ít dữ liệu được gán nhãn
- Ngoài việc tính toán khoảng cách từ một điểm dữ liệu kiểm tra đến tất cả các điểm trong tập huấn luyện có một số thuật toán khác giúp tăng tốc việc tìm kiếm này như K-D tree và Ball tre

Tài liệu tham khảo

[1] https://machinelearningcoban.com/2017/01/08/knn/
[2] https://www.kaggle.com/code/zarna99/iris-cross-validation
[3] https://machinelearningmastery.com/k-fold-cross-validation/
[4] https://www.geeksforgeeks.org/loocvleave-one-out-cross-validation-in-r-programming/

***Source Code

[1] Hệ thống phân loại hoa diên vĩ:
Modulel_AI_24_25/BTL2_Iris_flower_Clasification_KNN at main · haphucc/Modulel_AI_24_25 (github.com)
[2] Các phương pháp đánh giá: https://github.com/quangdinh17th/Iris_flower.git