TRƯỜNG ĐẠI HỌC THỦY LỢI

**KHOA CÔNG NGHỆ THÔNG TIN**

A blue and white logo

Description automatically generated with low confidence

**BÀI TẬP LỚN**

HỌC PHẦN: KHAI PHÁ DỮ LIỆU

**ĐỀ TÀI:   
Xây dựng hệ thống phát hiện và cảnh báo động vật xâm nhập khu nông nghiệp sử dụng các thuật toán Random Forest, K-Nearest Neighbors và Naive Bayes**

Giáo viên hướng dẫn: Lê Thị Tú Kiên

Nhóm thực hiện: 2

Các thành viên:

1. Lê Hoàn(nhóm trưởng), lớp 63CNTT.VA

2. Trần Đình Trung, lớp 63CNTT.VA

3. Nguyễn Văn Tuấn, lớp 63CNTT.VA

4. Nguyễn Đức Hiếu, lớp 63CNTT.VA

5. Phạm Ngọc Vinh, lớp 63CNTT.VA

**Hà Nội, năm 2025**

# MỤC LỤC

[MỤC LỤC 2](#_heading=h.gjdgxs)

[**Bảng phân chia công việc 3**](#_heading=h.30j0zll)

[I. Tổng quan và lý thuyết 4](#_heading=h.1fob9te)

[A. Giới thiệu về các thuật toán Classification 4](#_heading=h.2et92p0)

[**1. Thuật toán Random Forest 4**](#_heading=h.tyjcwt)

[a. Cách thức hoạt động của Random Forest: 4](#_heading=h.3dy6vkm)

[b. Random Forest - Sức mạnh của sự "ngẫu nhiên" và "tổng hợp": 5](#_heading=h.35nkun2)

[c. Ưu điểm của Random Forest: 6](#_heading=h.2xcytpi)

[d. Nhược điểm của Random Forest: 6](#_heading=h.1ci93xb)

[e. Các tham số quan trọng của Random Forest: 6](#_heading=h.2bn6wsx)

[f. Ứng dụng của Random Forest: 6](#_heading=h.qsh70q)

[**2. Thuật toán K-Nearest Neighbors: 6**](#_heading=h.3as4poj)

[a. Khái niệm: 6](#_heading=h.49x2ik5)

[b. Cách hoạt động: 6](#_heading=h.2p2csry)

[c. Các hàm đo khoảng cách (Distance Metrics): 8](#_heading=h.147n2zr)

[d. Ưu điểm của KNN: 8](#_heading=h.3o7alnk)

[e. Nhược điểm của KNN: 8](#_heading=h.23ckvvd)

[f. Cách chọn giá trị K: 8](#_heading=h.ihv636)

[g. Ứng dụng của KNN: 8](#_heading=h.4bvk7pj)

[**3. Thuật toán Naive Bayes Classifier: 9**](#_heading=h.2r0uhxc)

[a. Định lý Bayes: 9](#_heading=h.1664s55)

[b. Cách thức hoạt động: 9](#_heading=h.3q5sasy)

[c. Các loại Naive Bayes Classifiers: 10](#_heading=h.25b2l0r)

[d. Ưu điểm của Naive Bayes: 10](#_heading=h.kgcv8k)

[e. Nhược điểm của Naive Bayes: 10](#_heading=h.34g0dwd)

[f. Ứng dụng của Naive Bayes: 10](#_heading=h.1jlao46)

[II. Mô tả bài toán 11](#_heading=h.43ky6rz)

[A. Lý do chọn đề tài 11](#_heading=h.2iq8gzs)

[B. Mô tả bài toán 11](#_heading=h.xvir7l)

[C. Quy trình thực hiện 12](#_heading=h.3hv69ve)

[D. Phân tích dữ liệu thô 16](#_heading=h.1x0gk37)

[III. Tiền xử lý dữ liệu 16](#_heading=h.4h042r0)

[A. Tiền xử lý dữ liệu 16](#_heading=h.2w5ecyt)

[1. Tiền xử lý dữ liệu 16](#_heading=h.1baon6m)

[a) Làm sạch dữ liệu: 16](#_heading=h.3vac5uf)

[b) Tích hợp dữ liệu: 16](#_heading=h.2afmg28)

[c) Biến đổi dữ liệu: 17](#_heading=h.pkwqa1)

[2. Phân tích dữ liệu 17](#_heading=h.39kk8xu)

[3. Phân tách dữ liệu 17](#_heading=h.48pi1tg)

[IV. Xây dựng mô hình và kết quả thực hiện 18](#_heading=h.2nusc19)

[A. Xây dựng cây Random Forest 18](#_heading=h.1302m92)

[B. Xây dựng mô hình Naive Bayes Classification 22](#_heading=h.3mzq4wv)

[C. Xây dựng mô hình K-Nearest Neighbors 34](#_heading=h.2250f4o)

[**KẾT LUẬN 39**](#_heading=h.haapch)

[**TÀI LIỆU THAM KHẢO 40**](#_heading=h.319y80a)

# **Bảng phân chia công việc**

|  |  |
| --- | --- |
| **Họ và tên** | **Nhiệm vụ** |
| Lê Hoàn | Lý thuyết và báo cáo |
| Phạm Ngọc Vinh | Tiền xử lý dữ liệu |
| Nguyễn Đức Hiếu | Xây dựng thuật toán Naive Bayes Classification |
| Trần Đình Trung | Xây dựng thuật toán K-Nearest Neighbors |
| Nguyễn Văn Tuấn | Xây dựng thuật toán Random Forest |

# 

# Tổng quan và lý thuyết

## Giới thiệu về các thuật toán Classification

Phân loại (Classification) là một trong những bài toán quan trọng và phổ biến nhất trong học máy (Machine Learning) và khai phá dữ liệu (Data Mining). Mục tiêu của bài toán phân loại là xây dựng một mô hình có khả năng dự đoán nhãn (label) hoặc lớp (class) của một đối tượng dựa trên các thuộc tính (features) của đối tượng đó. Ví dụ, phân loại email là spam hay không spam, phân loại ảnh là chó hay mèo, phân loại khách hàng có khả năng rời bỏ dịch vụ hay không, v.v.

Có rất nhiều thuật toán phân loại khác nhau, mỗi thuật toán có những ưu điểm và nhược điểm riêng, phù hợp với các loại dữ liệu và bài toán khác nhau. Trong bản báo cáo này thì nhóm em sẽ trình bày về các thuật toán: Random Forest, K-Nearest Neighbors (KNN) và Naive Bayes.

### **Thuật toán Random Forest**

Random Forest là một thuật toán học máy thuộc nhóm Ensemble Learning (học tập hợp). Nó hoạt động bằng cách xây dựng nhiều cây quyết định (Decision Tree) trong quá trình huấn luyện và đưa ra kết quả dự đoán cuối cùng dựa trên tổ hợp (bỏ phiếu hoặc lấy trung bình) kết quả của từng cây. Mỗi cây quyết định được huấn luyện trên một tập dữ liệu con được lấy mẫu ngẫu nhiên có hoàn lại (bagging) từ tập dữ liệu huấn luyện gốc. Thêm vào đó, tại mỗi nút của cây quyết định, thay vì sử dụng tất cả các thuộc tính, Random Forest chỉ chọn ngẫu nhiên một tập con thuộc tính để tìm ra điều kiện phân chia tốt nhất.

#### Cách thức hoạt động của Random Forest:

* B1: Tạo các tập dữ liệu con (Bootstrap Samples)
  + Random Forest bắt đầu bằng cách tạo ra nhiều tập dữ liệu con (gọi là **bootstrap samples**) từ tập dữ liệu huấn luyện ban đầu.
  + Quá trình này được thực hiện bằng cách **lấy mẫu ngẫu nhiên có hoàn lại (sampling with replacement)** từ tập dữ liệu gốc. Điều này có nghĩa là:
    - Mỗi tập dữ liệu con có cùng kích thước với tập dữ liệu gốc.
    - Một mẫu dữ liệu (một hình ảnh trong trường hợp của bạn) có thể xuất hiện nhiều lần trong cùng một tập dữ liệu con.
    - Một số mẫu dữ liệu trong tập dữ liệu gốc có thể không xuất hiện trong tập dữ liệu con.
* B2: Xây dựng cây quyết định (Decision Tree Construction)
  + Trên mỗi tập dữ liệu con, Random Forest sẽ xây dựng một **cây quyết định** độc lập.
  + **Cây quyết định** là một cấu trúc dạng cây, trong đó:
    - Nút gốc (root node): Đại diện cho toàn bộ tập dữ liệu.
    - Nút trong (internal nodes): Đại diện cho một thuộc tính (feature) được chọn để kiểm tra.
    - Nhánh (branches): Đại diện cho các giá trị có thể có của thuộc tính ở nút cha.
  + Quá trình xây dựng cây quyết định:
    - Bắt đầu từ nút gốc.
    - Tại mỗi nút, chọn ngẫu nhiên một tập con các thuộc tính từ tất cả các thuộc tính (đặc trưng) có sẵn. (Random Feature Selection)
    - Tìm ra thuộc tính tốt nhất trong tập con này để phân chia dữ liệu. Thuộc tính tốt nhất được chọn dựa trên một tiêu chí nào đó, ví dụ như độ lợi thông tin (Information Gain) hoặc chỉ số Gini (Gini Index). Mục đích là để tạo ra các nút con có độ thuần khiết (purity) cao hơn, nghĩa là các mẫu dữ liệu trong cùng một nút con có xu hướng thuộc cùng một lớp.
    - Tạo các nút con dựa trên các giá trị có thể có của thuộc tính tốt nhất.
    - Lặp lại quá trình trên cho từng nút con cho đến khi đạt được một trong các điều kiện dừng, ví dụ như:
    - Tất cả các mẫu dữ liệu trong nút thuộc cùng một lớp.
    - Đạt đến độ sâu tối đa của cây.
    - Số lượng mẫu dữ liệu trong nút nhỏ hơn một ngưỡng tối thiểu.
* B3: Đưa ra dự đoán (Prediction)
  + Khi có một mẫu dữ liệu mới cần dự đoán (một hình ảnh mới trong trường hợp của bạn):
    - Cho mẫu dữ liệu đó đi qua **tất cả các cây quyết định** trong rừng.
    - Mỗi cây quyết định sẽ đưa ra một dự đoán về nhãn lớp của mẫu dữ liệu đó.
  + **Đối với bài toán phân loại (Classification):** Random Forest sử dụng phương pháp **biểu quyết đa số (majority voting)** để đưa ra dự đoán cuối cùng. Nhãn lớp được dự đoán nhiều nhất bởi các cây quyết định sẽ được chọn làm nhãn lớp cuối cùng cho mẫu dữ liệu mới.
  + **Đối với bài toán hồi quy (Regression):** Random Forest sử dụng phương pháp **trung bình cộng (averaging)**. Giá trị dự đoán cuối cùng là trung bình cộng các giá trị dự đoán của tất cả các cây.
* **Tóm tắt các đặc điểm quan trọng của Random Forest:**
  + Bagging (Bootstrap Aggregating): Tạo nhiều tập dữ liệu con bằng cách lấy mẫu ngẫu nhiên có hoàn lại từ tập dữ liệu gốc.
  + Random Feature Selection: Tại mỗi nút của cây quyết định, chỉ chọn ngẫu nhiên một tập con các thuộc tính để tìm ra thuộc tính tốt nhất để phân chia.
  + Ensemble Learning: Kết hợp kết quả dự đoán của nhiều cây quyết định để đưa ra kết quả cuối cùng.
  + Voting (Classification) / Averaging (Regression): Cách thức kết hợp kết quả dự đoán của các cây.

#### Random Forest - Sức mạnh của sự "ngẫu nhiên" và "tổng hợp":

Random Forest là một thuật toán ensemble learning (học máy tổ hợp), kết hợp nhiều cây quyết định để đưa ra dự đoán. Mỗi cây quyết định trong Random Forest được huấn luyện trên một tập dữ liệu con được lấy mẫu ngẫu nhiên từ tập dữ liệu huấn luyện ban đầu (bagging). Ngoài ra, tại mỗi nút, thay vì chọn thuộc tính tốt nhất trong tất cả các thuộc tính để chia, Random Forest chỉ chọn ngẫu nhiên một tập con các thuộc tính. Việc này giúp giảm sự tương quan giữa các cây, tăng tính đa dạng và cải thiện độ chính xác của mô hình. Kết quả dự đoán của Random Forest được tổng hợp từ kết quả dự đoán của các cây quyết định thành phần, thường bằng cách lấy trung bình (đối với bài toán hồi quy) hoặc bỏ phiếu (đối với bài toán phân loại).

#### Ưu điểm của Random Forest:

* Độ chính xác cao: Random Forest thường cho độ chính xác cao hơn so với các cây quyết định đơn lẻ.
* Khả năng xử lý dữ liệu nhiễu tốt: Random Forest ít bị ảnh hưởng bởi dữ liệu nhiễu và ngoại lai.
* Khả năng xử lý dữ liệu thiếu tốt: Random Forest có thể xử lý dữ liệu thiếu bằng cách sử dụng các giá trị trung bình hoặc trung vị.
* Ít bị overfitting: Do sử dụng kỹ thuật bagging và chọn ngẫu nhiên thuộc tính, Random Forest ít bị overfitting hơn so với các cây quyết định đơn lẻ.
* Có thể đánh giá tầm quan trọng của các thuộc tính.

#### Nhược điểm của Random Forest:

* Khó diễn giải: Random Forest là một mô hình "hộp đen", khó diễn giải hơn so với các cây quyết định đơn lẻ.
* Tốn thời gian và bộ nhớ: Việc huấn luyện Random Forest tốn nhiều thời gian và bộ nhớ hơn so với việc huấn luyện một cây quyết định đơn lẻ.

#### Các tham số quan trọng của Random Forest:

* n\_estimators: Số lượng cây quyết định trong rừng.
* max\_depth: Độ sâu tối đa của cây.
* min\_samples\_split: Số lượng mẫu tối thiểu để chia một nút.
* min\_samples\_leaf: Số lượng mẫu tối thiểu tại một nút lá.
* max\_features: Số lượng thuộc tính tối đa được xem xét để chia tại mỗi nút.

#### Ứng dụng của Random Forest:

* Phân loại văn bản, hình ảnh, âm thanh.
* Phát hiện gian lận.
* Chẩn đoán y tế.
* Dự đoán rủi ro tín dụng.

### **Thuật toán K-Nearest Neighbors:**

#### Khái niệm:

KNN là một thuật toán phân loại dựa trên độ tương đồng, không cần tham số (non-parametric), và "lười biếng" (lazy learning). KNN không xây dựng một mô hình tường minh từ dữ liệu huấn luyện, mà lưu trữ toàn bộ dữ liệu huấn luyện. Khi cần dự đoán nhãn của một đối tượng mới, KNN tìm kiếm K đối tượng gần nhất (hàng xóm) trong tập dữ liệu huấn luyện dựa trên một độ đo khoảng cách (ví dụ: khoảng cách Euclidean), và sau đó gán nhãn cho đối tượng mới dựa trên nhãn của K hàng xóm này (thường bằng cách lấy đa số).

#### Cách hoạt động:

* B1: Tính toán khoảng cách giữa đối tượng mới và tất cả các đối tượng trong tập dữ liệu huấn luyện.
  + KNN **không thực sự "học"** một mô hình từ dữ liệu huấn luyện như các thuật toán khác (ví dụ: như tạo ra các hệ số trong hồi quy tuyến tính).
  + Thay vào đó, KNN **lưu trữ toàn bộ** tập dữ liệu huấn luyện, bao gồm các đặc trưng (features) và nhãn (labels) tương ứng của từng điểm dữ liệu.
* B2: Chọn ra K đối tượng gần nhất với đối tượng mới.
  + Khi có một điểm dữ liệu mới cần phân loại (điểm truy vấn - query point), KNN sẽ tính toán **khoảng cách** từ điểm dữ liệu mới này đến **tất cả** các điểm dữ liệu trong tập huấn luyện.
  + **Khoảng cách** là thước đo sự tương đồng giữa hai điểm dữ liệu. Điểm dữ liệu càng gần nhau thì càng giống nhau.
  + Có nhiều **hàm đo khoảng cách (distance metrics)** khác nhau có thể được sử dụng, phổ biến nhất là:
    - **Euclidean Distance (Khoảng cách Euclid):**
      * + d(x, y) = sqrt(sum((x\_i - y\_i)^2))
      * Là khoảng cách "đường chim bay" giữa hai điểm trong không gian.
      * Thích hợp cho dữ liệu liên tục.
    - **Manhattan Distance (Khoảng cách Manhattan):**
      * + d(x, y) = sum(|x\_i - y\_i|)
      * Là tổng trị tuyệt đối của hiệu giữa các tọa độ tương ứng của hai điểm.
      * Thích hợp cho dữ liệu liên tục và khi các chiều không đồng nhất.
    - **Minkowski Distance (Khoảng cách Minkowski):**
      * + d(x, y) = (sum(|x\_i - y\_i|^p))^(1/p)
      * Là trường hợp tổng quát của Euclide và Manhattan (với p=1 là Manhattan, p=2 là Euclide).
      * Tổng quát hóa của Euclide và Manhattan.
    - **Cosine Similarity (Độ tương đồng Cosine):**
      * + similarity(x, y) = (x . y) / (||x|| \* ||y||)
      * Đo góc giữa hai vector, phù hợp khi hướng quan trọng hơn độ lớn.
      * Thích hợp cho dữ liệu dạng văn bản.
    - **Hamming Distance (Khoảng cách Hamming):**
      * Đếm số lượng vị trí mà các giá trị tương ứng khác nhau giữa hai chuỗi.
      * Thích hợp cho dữ liệu phân loại (categorical).
* B3: Xác định nhãn của đối tượng mới dựa trên nhãn của K đối tượng gần nhất:
  + Sau khi tính toán khoảng cách từ điểm dữ liệu mới đến tất cả các điểm dữ liệu trong tập huấn luyện, KNN sẽ chọn ra **K** điểm dữ liệu có khoảng cách **nhỏ nhất** (gần nhất).
  + **K** là một tham số (hyperparameter) cần được chọn trước, và giá trị của K ảnh hưởng đến kết quả phân loại.
* B4: Dự đoán nhãn(label Prediction)
  + **Đối với bài toán phân loại (Classification):**
    - KNN sử dụng phương pháp **biểu quyết đa số (majority voting)**. Nhãn lớp xuất hiện **nhiều nhất** trong số K láng giềng gần nhất sẽ được gán cho điểm dữ liệu mới.
    - Ví dụ: Nếu K=5 và trong số 5 láng giềng gần nhất có 3 láng giềng thuộc lớp "chuột", 1 láng giềng thuộc lớp "ếch", và 1 láng giềng thuộc lớp "châu chấu", thì điểm dữ liệu mới sẽ được dự đoán thuộc lớp "chuột".
  + **Đối với bài toán hồi quy (Regression):**
    - KNN tính **giá trị trung bình (average)** của các giá trị (target values) của K láng giềng gần nhất để làm giá trị dự đoán cho điểm dữ liệu mới.
* **Tóm tắt các đặc điểm quan trọng của KNN:**
  + **Non-parametric:** Không đưa ra giả định về phân phối của dữ liệu.
  + **Instance-based:** Dựa trên việc so sánh trực tiếp với các điểm dữ liệu trong tập huấn luyện.
  + **Lazy learning:** Không "học" một mô hình cụ thể trong giai đoạn huấn luyện, mà chỉ lưu trữ dữ liệu. Việc tính toán chỉ diễn ra khi có yêu cầu dự đoán.

#### Các hàm đo khoảng cách (Distance Metrics):

* Khoảng cách Euclidean: Khoảng cách đường chim bay giữa hai điểm trong không gian nhiều chiều.
* Khoảng cách Manhattan: Tổng trị tuyệt đối của hiệu giữa các tọa độ tương ứng của hai điểm.
* Khoảng cách Minkowski: Tổng quát hóa của khoảng cách Euclide và Manhattan.
* Khoảng cách Chebyshev: Khoảng cách lớn nhất giữa các tọa độ tương ứng của hai điểm.

#### Ưu điểm của KNN:

* Đơn giản, dễ hiểu và dễ cài đặt.
* Không cần giả định về phân phối của dữ liệu.
* Có thể thích ứng với dữ liệu mới dễ dàng (chỉ cần thêm dữ liệu mới vào tập dữ liệu huấn luyện).

#### Nhược điểm của KNN:

* Độ phức tạp tính toán cao khi dự đoán, đặc biệt với tập dữ liệu lớn.
* Nhạy cảm với dữ liệu nhiễu và các thuộc tính không liên quan.
* Cần lựa chọn giá trị K phù hợp.
* Khó khăn khi xử lý dữ liệu thiếu.

#### Cách chọn giá trị K:

* Thử nghiệm với các giá trị K khác nhau và đánh giá hiệu suất của mô hình trên tập dữ liệu kiểm tra (validation set).
* Sử dụng các kỹ thuật như cross-validation để chọn K.
* Thông thường, K nên là một số lẻ để tránh trường hợp hòa phiếu.
* K quá nhỏ có thể dẫn đến overfitting, K quá lớn có thể dẫn đến underfitting.

#### Ứng dụng của KNN:

* Hệ thống gợi ý (recommendation systems).
* Nhận dạng ký tự quang học (OCR).
* Phân loại văn bản.
* Phát hiện xâm nhập mạng.

### **Thuật toán Naive Bayes Classifier:**

Naive Bayes là một thuật toán phân loại học máy xác suất dựa trên Định lý Bayes. Nó được gọi là "Naive" (ngây thơ) vì nó giả định rằng các thuộc tính (đặc trưng) là độc lập thống kê với nhau khi biết giá trị của nhãn lớp. Mặc dù giả định này hiếm khi đúng trong thực tế, nhưng Naive Bayes vẫn cho kết quả khá tốt và được sử dụng rộng rãi, đặc biệt trong các bài toán phân loại văn bản. Khi huấn luyện, Naive Bayes sẽ tính toán xác suất tiên nghiệm (prior probability) của từng lớp và xác suất có điều kiện (conditional probability) của từng thuộc tính trong từng lớp. Khi cần phân loại một điểm dữ liệu mới, Naive Bayes sẽ tính toán xác suất hậu nghiệm (posterior probability) của từng lớp dựa trên các thuộc tính của điểm dữ liệu đó và chọn ra lớp có xác suất cao nhất.

#### Định lý Bayes:

* Định lý Bayes là một công thức toán học mô tả xác suất của một sự kiện dựa trên kiến thức tiên nghiệm về các điều kiện có thể liên quan đến sự kiện đó. Công thức: P(A|B) = [P(B|A) \* P(A)] / P(B). Trong đó:
  + P(A|B): Xác suất xảy ra sự kiện A khi biết sự kiện B đã xảy ra (xác suất hậu nghiệm).
  + P(B|A): Xác suất xảy ra sự kiện B khi biết sự kiện A đã xảy ra (likelihood).
  + P(A): Xác suất xảy ra sự kiện A (xác suất tiên nghiệm).
  + P(B): Xác suất xảy ra sự kiện B.

#### Cách thức hoạt động:

* B1: Áp dụng Định lý Bayes cho Phân loại
  + Trong bài toán phân loại, chúng ta muốn tìm xác suất một điểm dữ liệu X = (x1, x2, ..., xn) (với x1, x2, ..., xn là các giá trị của các thuộc tính) thuộc về một lớp C nào đó.
  + Áp dụng định lý Bayes, ta có:
    - P(C|X) = [P(X|C) \* P(C)] / P(X)
  + Trong đó:
    - P(C|X): Xác suất điểm dữ liệu X thuộc về lớp C (posterior probability) - đây là xác suất chúng ta cần tính.
    - P(X|C): Xác suất quan sát thấy các thuộc tính X khi biết điểm dữ liệu thuộc về lớp C (likelihood).
    - P(C): Xác suất của lớp C (prior probability) - xác suất này có thể được ước lượng từ tập dữ liệu huấn luyện.
    - P(X): Xác suất quan sát thấy các thuộc tính X (evidence) - thường được bỏ qua trong tính toán vì nó là hằng số đối với tất cả các lớp.
* B2: Giả định "Naive" (Độc lập có điều kiện)
  + Để đơn giản hóa việc tính toán P(X|C), Naive Bayes đưa ra giả định "ngây thơ" rằng các thuộc tính x1, x2, ..., xn là độc lập có điều kiện với nhau khi biết lớp C.
  + Điều này có nghĩa là: P(X|C) = P(x1|C) \* P(x2|C) \* ... \* P(xn|C)
    - Giả định này hiếm khi đúng trong thực tế, nhưng nó giúp việc tính toán trở nên dễ dàng hơn và Naive Bayes vẫn cho kết quả tốt trong nhiều trường hợp.
* B3: Huấn luyện mô hình (Training)
  + Trong giai đoạn huấn luyện, Naive Bayes ước lượng các xác suất P(C) và P(xi|C) từ tập dữ liệu huấn luyện.
  + P(C) (xác suất tiên nghiệm của lớp C): Được ước lượng bằng cách tính tỷ lệ số lượng điểm dữ liệu thuộc lớp C trên tổng số điểm dữ liệu trong tập huấn luyện.
  + P(xi|C) (xác suất có điều kiện của thuộc tính xi khi biết lớp C): Cách tính P(xi|C) phụ thuộc vào loại thuộc tính xi (liên tục hay rời rạc) và loại Naive Bayes được sử dụng (Gaussian, Multinomial, Bernoulli).
* B4: Dự đoán nhãn (Prediction):
  + Khi có một điểm dữ liệu mới X cần phân loại:
    - Tính P(X|C) cho từng lớp C sử dụng giả định độc lập thuộc tính:
      * P(X|C) = P(x1|C) \* P(x2|C) \* ... \* P(xn|C).
    - Tính P(C|X) cho từng lớp C sử dụng công thức Bayes (thường bỏ qua P(X) vì nó là hằng số):
      * P(C|X) ∝ P(X|C) \* P(C).
    - Chọn lớp C có P(C|X) cao nhất làm lớp dự đoán cho điểm dữ liệu X.

#### Các loại Naive Bayes Classifiers:

* Gaussian Naive Bayes: Giả định các thuộc tính liên tục tuân theo phân phối chuẩn (Gaussian).
* Multinomial Naive Bayes: Thích hợp cho dữ liệu rời rạc, chẳng hạn như tần suất xuất hiện của các từ trong văn bản.
* Bernoulli Naive Bayes: Thích hợp cho dữ liệu nhị phân (0/1), chẳng hạn như sự xuất hiện/vắng mặt của các từ trong văn bản.

#### Ưu điểm của Naive Bayes:

* Đơn giản, dễ cài đặt và huấn luyện nhanh.
* Hiệu quả với dữ liệu chiều cao (high-dimensional data), chẳng hạn như văn bản.
* Có thể xử lý dữ liệu thiếu.
* Ít bị ảnh hưởng bởi các thuộc tính không liên quan.

#### Nhược điểm của Naive Bayes:

* Giả định độc lập giữa các thuộc tính thường không đúng trong thực tế, có thể ảnh hưởng đến độ chính xác của mô hình.
* Nhạy cảm với cách biểu diễn dữ liệu.

#### Ứng dụng của Naive Bayes:

* Phân loại thư rác (spam filtering).
* Phân tích cảm xúc (sentiment analysis).
* Phân loại văn bản.
* Chẩn đoán y tế.

# Mô tả bài toán

## Lý do chọn đề tài

Trong bối cảnh an ninh và an toàn tại các khu nông nghiệp ngày càng được chú trọng, việc giám sát và bảo vệ các khu vực này khỏi sự xâm nhập trái phép của động vật trở thành một nhiệm vụ quan trọng. Sự xâm nhập của động vật hoang dã hoặc thú nuôi thả rông không chỉ gây thiệt hại về cây trồng, vật nuôi, ảnh hưởng đến năng suất và sản lượng nông nghiệp mà còn tiềm ẩn nguy cơ lây lan dịch bệnh, mất an toàn cho con người và chính các loài động vật. Đặc biệt, các khu nông nghiệp thường có diện tích rộng lớn, địa hình phức tạp, khó khăn trong việc tuần tra, giám sát, khiến cho việc bảo vệ bằng các biện pháp truyền thống dựa trên sức người gặp nhiều hạn chế về chi phí, nhân lực và hiệu quả hoạt động. Do đó, việc ứng dụng các thuật toán học máy như Random Forest, K-Nearest Neighbors và Naive Bayes để phát hiện và cảnh báo sớm sự xâm nhập của động vật là một giải pháp thiết thực và hiệu quả. Đề tài "Xây dựng hệ thống phát hiện và cảnh báo động vật xâm nhập khu nông nghiệp sử dụng các thuật toán Random Forest, K-Nearest Neighbors và Naive Bayes" không chỉ là sự kết hợp giữa lý thuyết và thực tiễn mà còn là cơ hội giúp chúng em nghiên cứu và phát triển kỹ năng giải quyết vấn đề cụ thể, ứng dụng kiến thức đã học vào một bài toán thực tế, góp phần nâng cao an ninh, an toàn, bảo vệ mùa màng, vật nuôi và hướng tới sự phát triển bền vững của ngành nông nghiệp trong kỷ nguyên số. Hơn nữa, việc phát triển hệ thống này còn góp phần bảo vệ động vật khỏi những nguy hiểm tiềm tàng khi chúng xâm nhập vào các khu vực canh tác.

## Mô tả bài toán

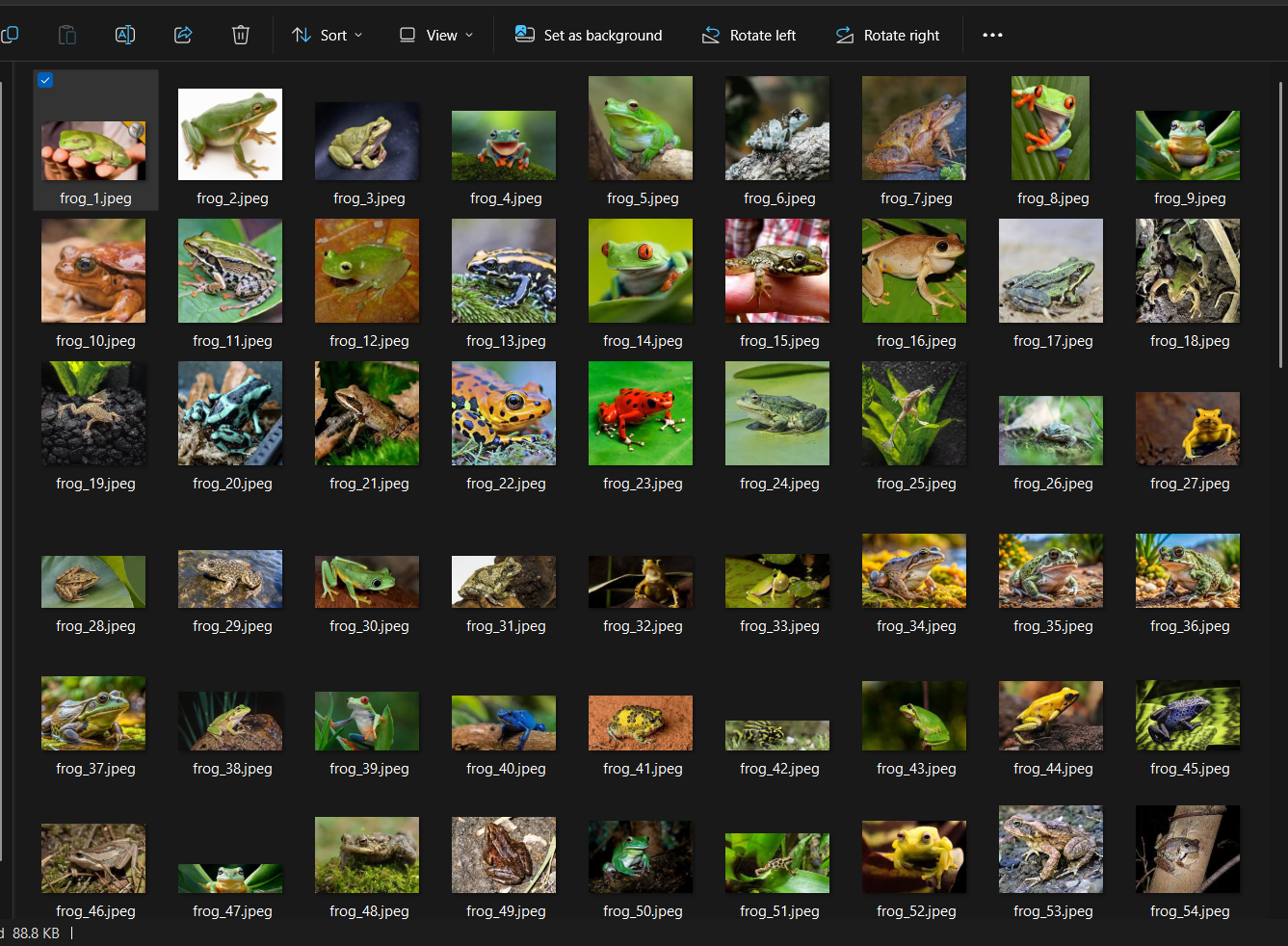
Bài toán đề xuất tập trung vào việc xây dựng một hệ thống phát hiện và cảnh báo các loài động vật gây hại trong khu nông nghiệp, thông qua việc áp dụng các thuật toán học máy: **Random Forest, K-Nearest Neighbors và Naive Bayes**. Hệ thống sẽ sử dụng tập dữ liệu hình ảnh/video thu thập từ camera giám sát để phân loại, xác định sự hiện diện và loài động vật xuất hiện trong khung hình. Từ đó, đưa ra cảnh báo về mức độ có hại của các loài động vật này đối với khu nông nghiệp.

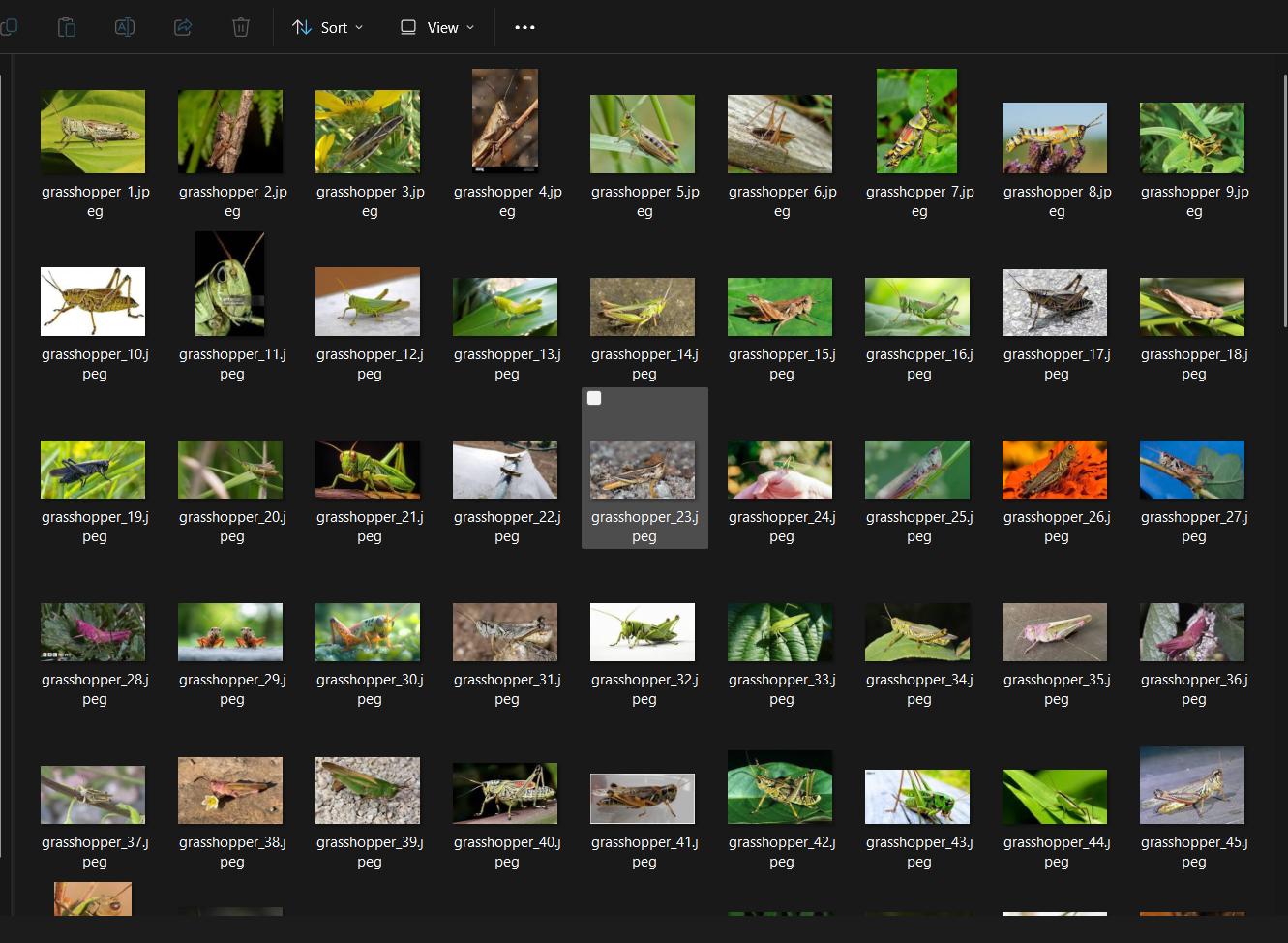
* **Đầu vào:** Tập dữ liệu hình ảnh/video thu thập từ camera giám sát tại khu nông nghiệp cần bảo vệ, đã được gán nhãn (labeled data) với thông tin về loài động vật (nếu có) trong ảnh/khung hình. Bài toán này tập trung vào 3 loài động vật chính: **chuột (rat), ếch (frog), và châu chấu (grasshopper)**. Các ảnh không có động vật sẽ được gán nhãn **"không có động vật" (none)**.
* **Đầu ra:**
  + Phát hiện sự hiện diện của động vật trong khu nông nghiệp được giám sát.
  + Phân loại loài động vật xâm nhập thành 3 loài: **chuột (rat), ếch (frog), châu chấu (grasshopper)**, hoặc **none** (không có động vật).
  + Xác định mức độ gây hại của từng loài:
    - **Có hại:** Chuột, Châu chấu.
    - **Không có hại (có lợi):** Ếch.
  + Thông tin cảnh báo (có thể bao gồm: thời gian, vị trí/camera phát hiện, hình ảnh/video về sự xâm nhập, **mức độ gây hại**) được gửi đến người quản lý/hệ thống giám sát của khu nông nghiệp.
* Mục tiêu cuối cùng của bài toán này là xây dựng một hệ thống sử dụng kết hợp các thuật toán **Random Forest, K-Nearest Neighbors và Naive Bayes** để tự động phát hiện, phân loại và cảnh báo khi có động vật xâm nhập vào khu nông nghiệp. Từ đó, phân loại mức độ có hại hay có lợi (không có hại) của các loài dựa trên kiến thức sinh học cơ bản(chuột và châu chấu có hại cho mùa màng, ếch có lợi vì ăn châu chấu). Việc này giúp nâng cao hiệu quả giám sát, hỗ trợ đưa ra quyết định phòng trừ kịp thời (ví dụ: tăng cường bẫy chuột, bảo vệ ếch), giảm thiểu chi phí nhân lực, đảm bảo an ninh, an toàn, và bảo vệ cây trồng, từ đó nâng cao năng suất và hiệu quả kinh tế cho khu nông nghiệp.

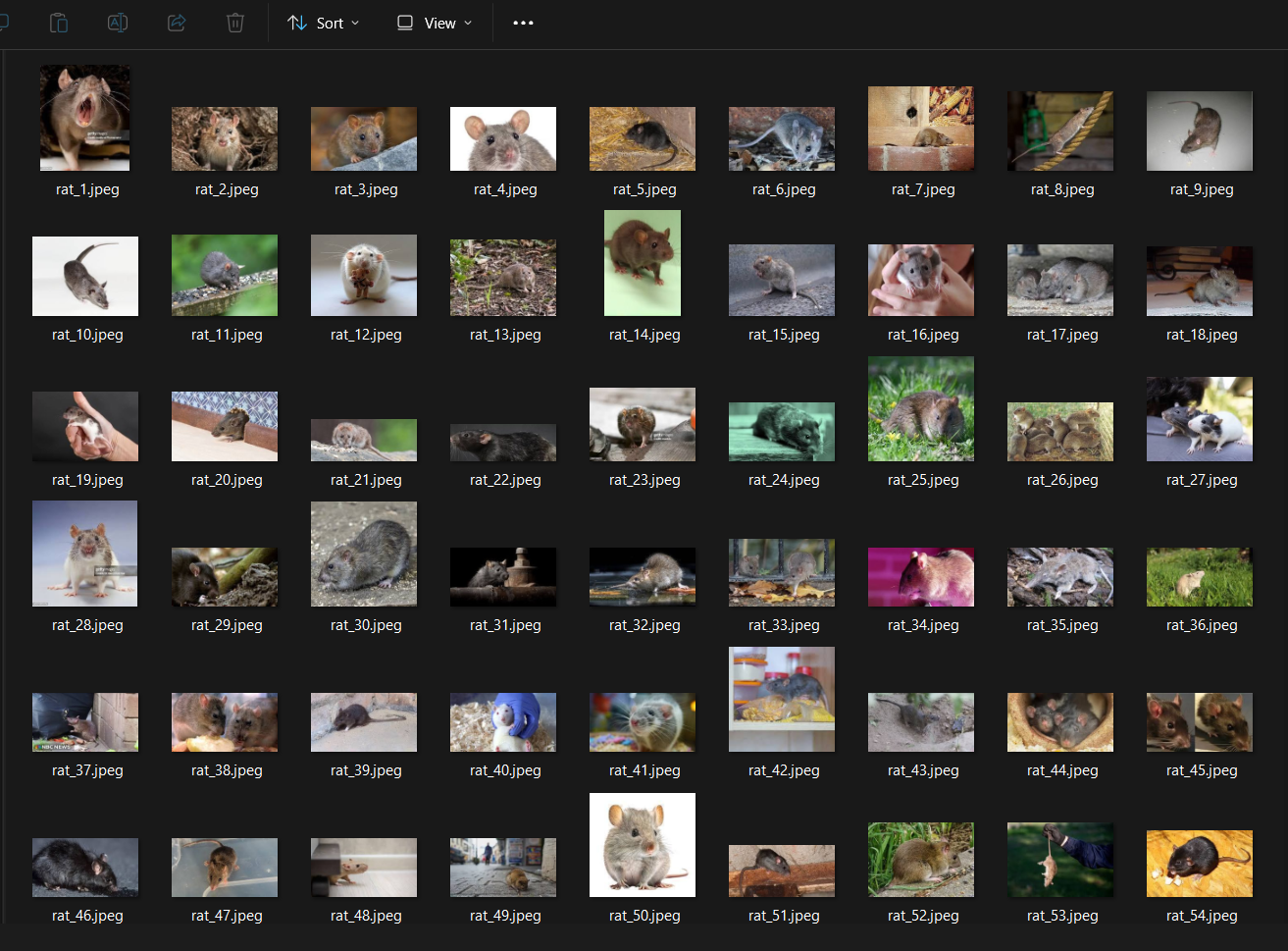
## Quy trình thực hiện

Quá trình xây dựng hệ thống phát hiện và cảnh báo động vật xâm nhập khu nông nghiệp bao gồm các bước chính sau:

* **B1: Thu thập dữ liệu:**
  + Sử dụng công cụ tìm kiếm hình ảnh như **Google Images** và **Bing Images** để tìm kiếm và tải về hình ảnh của các loài động vật: **ếch, chuột, châu chấu**.







* + **Gán nhãn cho dữ liệu:** Xác định loài động vật xuất hiện trong từng ảnh và gán nhãn tương ứng:
    - **frog:** cho hình ảnh ếch.
    - **rat:** cho hình ảnh chuột.
    - **grasshopper:** cho hình ảnh châu chấu.
  + Xây dựng tập dữ liệu: Sau khi thu thập và gán nhãn, tập dữ liệu sẽ được chia thành các tập: tập huấn luyện (training set), tập kiểm tra (test set).
* **B2: Tiền xử lý dữ liệu:**
  + Làm sạch dữ liệu:
    - Loại bỏ các hình ảnh bị mờ, nhiễu, không rõ ràng, hoặc bị che khuất.
    - Loại bỏ các hình ảnh không đúng loài cần tìm.
    - Cẩn thận loại bỏ các ảnh có watermark hoặc logo của nguồn ảnh.
    - Loại bỏ các hình ảnh trùng lặp.
  + Chuẩn hóa dữ liệu:
    - Điều chỉnh kích thước (resize) tất cả các hình ảnh về cùng một kích thước chuẩn.
    - Cân bằng sáng, tăng cường độ tương phản (nếu cần).
    - Chuẩn hóa các giá trị pixel (ví dụ: đưa về khoảng giá trị từ 0 đến 1).
* **B3: Trích xuất đặc trưng (Feature Extraction) và Biến đổi dữ liệu:**
  + **Sử dụng mạng CNN (Convolutional Neural Networks) pre-trained, cụ thể là ResNet, để trích xuất đặc trưng:**
  + **Lý do lựa chọn ResNet:** ResNet (Residual Network) là một kiến trúc mạng CNN sâu, nổi tiếng với khả năng học các đặc trưng phức tạp và đạt hiệu suất cao trong các tác vụ phân loại ảnh. ResNet sử dụng các **residual blocks** (khối phần dư) để giải quyết vấn đề **vanishing gradient** (đạo hàm biến mất) thường gặp trong các mạng nơ-ron sâu, cho phép huấn luyện các mạng có độ sâu lớn hơn (ví dụ: ResNet-50, ResNet-101, ResNet-152). Các mô hình ResNet đã được huấn luyện trước trên tập dữ liệu ImageNet (chứa hàng triệu hình ảnh thuộc 1000 lớp khác nhau) có khả năng trích xuất các đặc trưng tốt cho nhiều loại ảnh khác nhau.
  + **Cách thức sử dụng ResNet để trích xuất đặc trưng:**

1. **Loại bỏ lớp fully connected (fully connected layer) cuối cùng** của mô hình ResNet (lớp này được sử dụng để phân loại 1000 lớp của ImageNet).
2. **Coi phần còn lại của mạng ResNet (các lớp convolutional và pooling) như một bộ trích xuất đặc trưng cố định (fixed feature extractor).**
3. **Đưa hình ảnh đầu vào (sau khi đã tiền xử lý ở B2) qua mạng ResNet này để thu được các feature maps (bản đồ đặc trưng) ở các lớp trước lớp fully connected cuối cùng.** Thông thường, các lớp gần cuối mạng (ví dụ: lớp avg\_pool trước lớp fc trong ResNet) sẽ được sử dụng để trích xuất đặc trưng.
4. **Biến đổi các feature maps này thành vector đặc trưng** (ví dụ: bằng cách sử dụng global average pooling). Vector này sẽ đại diện cho hình ảnh đầu vào và được sử dụng cho các bước tiếp theo (huấn luyện mô hình phân loại).

* **B4: Huấn luyện mô hình và Phân tách dữ liệu (Training, Testing):**
  + **Số lượng mẫu cho mỗi loài:** **150 hình ảnh** (tổng cộng 450 hình ảnh cho 3 loài).
  + Phân tách dữ liệu: Chia tập dữ liệu thành 2 phần: tập huấn luyện (training set), tập kiểm tra (test set). Tỉ lệ phân chia: 80% training, 20% testing.
  + Huấn luyện mô hình:
    - Huấn luyện riêng biệt từng mô hình Random Forest, K-Nearest Neighbors và Naive Bayes trên tập dữ liệu huấn luyện (training set) đã được trích xuất đặc trưng.
    - Sử dụng các kỹ thuật như cross-validation để đánh giá và lựa chọn mô hình tối ưu.
  + Kết hợp các mô hình:
    - Xây dựng phương pháp kết hợp kết quả dự đoán từ 3 mô hình (ví dụ: voting, averaging, stacking)..

## Phân tích dữ liệu thô

* Dữ liệu lấy từ Google Images và Bing Images:
  + [Bing Images](https://www.bing.com/images/), <https://images.google.com/>
* Bộ dữ liệu cung cấp thông tin về hình ảnh của 3 loài động vật với 2 nhãn dữ liệu gồm:
  + “Có hại”: chuột, châu chấu.
  + “Không có hại”: ếch.
* Tổng quan dữ liệu:
  + Số lượng bản ghi của mỗi loài: 150 mẫu (tổng là 450).

# Tiền xử lý dữ liệu

## Tiền xử lý dữ liệu

### Tiền xử lý dữ liệu

#### Làm sạch dữ liệu:

*Bước này nhằm loại bỏ các dữ liệu nhiễu, không phù hợp, đảm bảo chất lượng dữ liệu đầu vào cho mô hình.*

* Loại bỏ hình ảnh lỗi:
  + Loại bỏ các hình ảnh bị hỏng, không thể mở, hoặc có định dạng không được hỗ trợ.
  + Loại bỏ các hình ảnh có kích thước quá nhỏ, độ phân giải quá thấp, không thể nhận dạng được động vật.
* Loại bỏ hình ảnh không liên quan:
  + Loại bỏ các hình ảnh không chứa động vật (chuột, ếch, châu chấu) hoặc chứa các đối tượng không liên quan đến bài toán (ví dụ: chỉ có cây cối, nhà cửa, con người,...). Bước này thực hiện thủ công.
* Xử lý hình ảnh mờ, nhòe, thiếu sáng:
  + Loại bỏ các hình ảnh quá mờ, nhòe, nhiễu, không thể nhận dạng được đối tượng.
* Xử lý ảnh có watermark:
  + Đối với các ảnh có watermark nhưng vẫn có thể nhận dạng rõ ràng các loài vật, và số lượng ảnh đủ lớn thì ta tiến hành loại bỏ watermark, logo bằng các công cụ chỉnh sửa ảnh hoặc các thư viện chuyên dụng.
  + Loại bỏ các ảnh có watermark lớn, che khuất phần lớn đối tượng, hoặc không thể loại bỏ watermark.

#### Tích hợp dữ liệu:

*Bước này tập hợp dữ liệu từ 2 nguồn: Google Images và Bing Images để tạo thành một tập dữ liệu thống nhất.*

* **Tổng hợp hình ảnh:** Tập hợp tất cả các hình ảnh đã thu thập từ Google Images và Bing Images vào cùng một thư mục.
* **Chuẩn hóa nhãn:** Đảm bảo rằng tất cả các hình ảnh đều được gán nhãn thống nhất theo quy tắc:
  + **rat:** cho hình ảnh chuột.
  + **frog:** cho hình ảnh ếch.
  + **grasshopper:** cho hình ảnh châu chấu.
  + **none:** cho hình ảnh không có động vật.

#### Biến đổi dữ liệu:

* Chuẩn hóa dữ liệu (Normalization):
  + Resize: Thay đổi kích thước tất cả các hình ảnh về cùng một kích thước chuẩn (ví dụ: 224x224 pixels) để phù hợp với đầu vào của mô hình ResNet.
  + Chuẩn hóa giá trị pixel: Đưa giá trị pixel của ảnh về khoảng [0, 1] hoặc [-1, 1] bằng cách chia cho 255. Việc này giúp mô hình huấn luyện nhanh hơn và ổn định hơn.

### Phân tích dữ liệu

Thống kê mô tả:

* **Tổng số lượng ảnh:** 450 ảnh cho 3 loài động vật: ếch, chuột, châu chấu
* **Số lượng ảnh cho mỗi lớp:** Sau khi làm sạch, có 150 ảnh cho mỗi loài
* **Phân bố số lượng ảnh giữa các lớp:**
  + Kiểm tra xem số lượng ảnh giữa các lớp có cân bằng hay không.
  + Vẽ biểu đồ histogram để trực quan hóa phân bố số lượng ảnh.
* **Kích thước ảnh:** Thống kê kích thước phổ biến của các hình ảnh sau khi resize.
* **Nhãn (label):**
  + **Có 2 nhãn được sử dụng: “có hại” và “không có hại”**
  + Đề cập đến việc gán nhãn **"có hại"** cho chuột, châu chấu và **"không có hại"** (có lợi) cho ếch.

### Phân tách dữ liệu

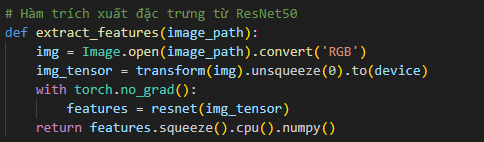
* **Phương pháp phân tách:** Chia tập dữ liệu thành 2 tập con: **tập huấn luyện (training set)** và **tập kiểm tra (test set)**.
* **Tỷ lệ phân chia:**
  + **80% cho tập huấn luyện (training set)**
  + **20% cho tập kiểm tra (test set)**
* **Cách thức thực hiện:**
  + Sử dụng hàm train\_test\_split trong thư viện scikit-learn để phân chia dữ liệu.
  + Đặt tham số random\_state để đảm bảo kết quả phân chia có thể tái lập được (khi chạy lại code sẽ cho ra cùng một kết quả phân chia).

# Xây dựng mô hình và kết quả thực hiện

## Xây dựng cây Random Forest

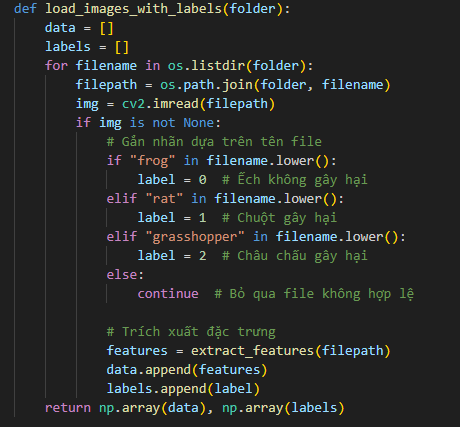
1. Ngôn ngữ: Python
2. Trình soạn thảo: Visual Studio Code
3. Các thư viện sử dụng:
   * Numpy: Xử lý mảng số liệu hiệu quả
   * Cv2 ( OpenCV): Dùng để xử lý hình ảnh ( đọc, resize, chuyển màu)
   * os: Xử lý đường dẫn và file hệ thống
   * sklearn: Sử dụng các mô hình học máy ( Random Forest ) và các công cụ khác nhau như chia dữ liệu, tính độ chính xác.
   * torch và torchvision: Thư viện học sâu và các mô hình pretrained để trích xuất đặc trưng từ ảnh.
4. Các bước xây dựng:

* Hàm trích xuất đặc trưng từ Resnet:



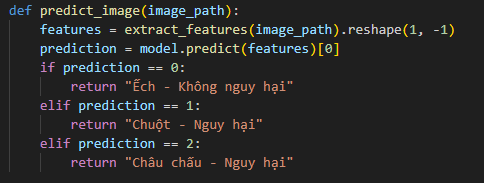
* Resnet được sử dụng để trích xuất đặc trưng từ ảnh ( bỏ qua lớp cuối, chỉ lấy đầu ra của lớp feature)

* Hàm load dữ liệu và trích xuất đặc trưng:



* Hàm này duyệt qua thư mục chứa ảnh, trích xuất đặc trưng cho từng ảnh và gán nhãn phù hợp (0: Ếch, 1: Chuột, 2: Châu chấu)

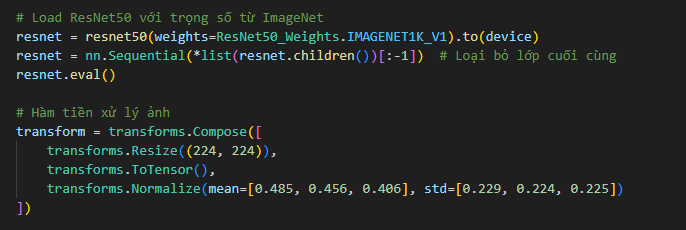
* Hàm dự đoán:



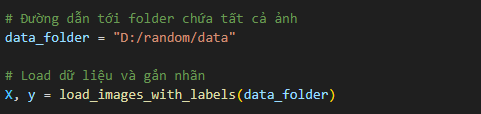
* Hàm này dùng để dự đoán loài của một ảnh và chỉ ra mức độ nguy hại.
* Thiết lập lại thiết bị:



* Chương trình tự động kiểm tra GPU có sẵn hay không. Nếu có, dùng GPU để tăng tốc độ xử lý.
* Tải mô hình Resnet và Pipeline tiền xử lý dữ liệu



* Resnet50: Mô hình ResNet50 pretrained trên ImageNet, được sử dụng để trích xuất đặc trưng từ ảnh.
* Transforms: Pipeline tiền xử lý ảnh, bao gồm resize, chuyển đổi sang tensor và chuẩn hóa theo chuẩn ImageNet.
* Load dữ liệu

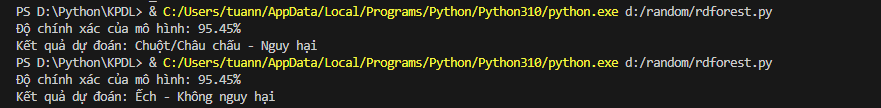


* Chương trình chạy qua thư mục chứa tất cả các ảnh của 3 loài và gán dữ liệu.
* Chia dữ liệu, huấn luyện mô hình và trình độ chính xác



* Dữ liệu được chia thành tập huấn luyện (80%) và tập kiểm tra (20%)
* Tôi chọn số lượng cây quyết định là 100 để giúp tăng độ chính xác của mô hình và số 42 là số điển hình trong lập trình
* Mô hình dự đoán trên tập kiểm tra (X\_test) và so sánh với nhãn thật sự (Y\_test) để tính độ chính xác.

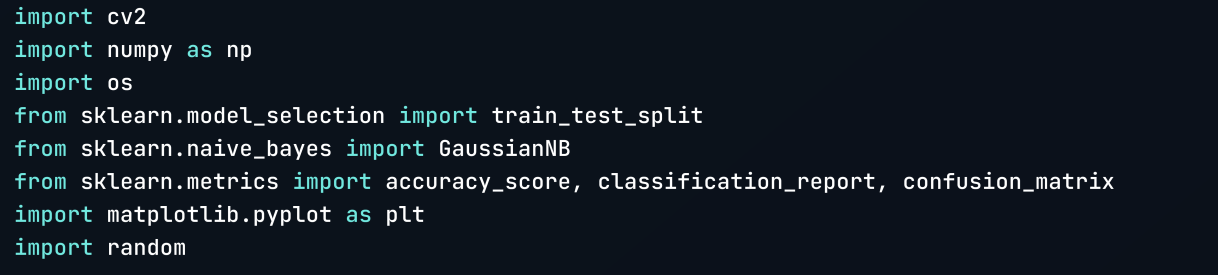
1. Kết quả chạy code



* Kết quả dự đoán chính xác cao tuy nhiên thời gian chạy khá lâu khoảng từ 50-60s

## Xây dựng mô hình Naive Bayes Classification

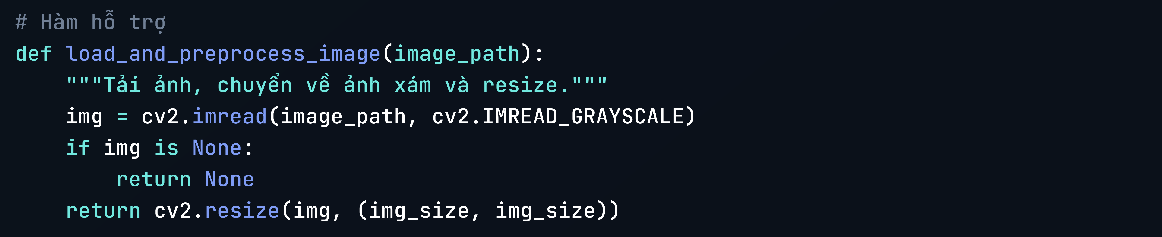
1. Ngôn ngữ: Python
2. Trình soạn thảo: Jupyter Notebook
3. Xây dựng chương trình:
   * 1. Chuẩn bị - Import thư viện:



* **cv2 (OpenCV):** Thư viện xử lý ảnh (tải ảnh, thay đổi kích thước, v.v.).
* **numpy:** Thư viện làm việc với mảng đa chiều (dùng để lưu trữ dữ liệu ảnh).
* **os:** Thư viện tương tác với hệ điều hành (duyệt file, thư mục).
* **train\_test\_split:** Hàm từ thư viện sklearn (scikit-learn) để chia dữ liệu thành tập huấn luyện và tập kiểm tra.
* **GaussianNB:** Mô hình phân loại Naive Bayes từ sklearn.
* **accuracy\_score, classification\_report, confusion\_matrix:** Các hàm đánh giá mô hình từ sklearn.
* **matplotlib.pyplot:** Thư viện để vẽ biểu đồ, hiển thị ảnh.
* **random:** Thư viện để tạo số ngẫu nhiên (dùng trong data augmentation)
  + 1. Cài đặt ban đầu:



* **data\_dir:** Đường dẫn đến thư mục chứa dữ liệu ảnh.
* **classes:** Tên các lớp cần phân loại.
* **img\_size:** Kích thước ảnh sau khi resize (ảnh sẽ được đưa về kích thước 256x256).
  + 1. Hàm hỗ trợ:



* **load\_and\_preprocess\_image(image\_path):**
  + Nhận đầu vào là đường dẫn ảnh.
  + Đọc ảnh bằng cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE) (đọc ảnh và chuyển về ảnh xám).
  + Kiểm tra xem ảnh có được tải thành công không (nếu img is None tức là tải không thành công).
  + Resize ảnh về kích thước img\_size x img\_size bằng cv2.resize().
  + Trả về ảnh đã được xử lý.
    1. Thu thập và tiền xử lý dữ liệu:



* **data:** Danh sách lưu trữ dữ liệu ảnh (mỗi ảnh là một vector 1D).
* **labels:** Danh sách lưu trữ nhãn tương ứng với mỗi ảnh (0: frog, 1: grasshopper, 2: rat).
* Vòng lặp for class\_name in classes: duyệt qua từng lớp ("frog", "grasshopper", "rat").
  + class\_dir = os.path.join(data\_dir, class\_name): Tạo đường dẫn đến thư mục của từng lớp.
  + Vòng lặp for image\_name in os.listdir(class\_dir): duyệt qua từng ảnh trong thư mục của lớp đó.
    - image\_path = os.path.join(class\_dir, image\_name): Tạo đường dẫn đầy đủ đến từng ảnh.
    - img = load\_and\_preprocess\_image(image\_path): Gọi hàm load\_and\_preprocess\_image để tải và tiền xử lý ảnh.
    - if img is not None: Nếu tải ảnh thành công:
      * data.append(augmented\_img.flatten()): Chuyển ảnh từ dạng 2D (64x64) thành vector 1D (4096) bằng flatten() và thêm vào danh sách data.
      * labels.append(classes.index(class\_name)): Lấy chỉ số của lớp (0, 1, hoặc 2) và thêm vào danh sách labels.
* data = np.array(data): Chuyển danh sách data thành mảng NumPy.
* labels = np.array(labels): Chuyển danh sách labels thành mảng NumPy.
  + 1. Tìm hiểu tập dữ liệu:



* In ra một số thông tin cơ bản về tập dữ liệu: số lượng ảnh, số lượng nhãn, kích thước mỗi ảnh sau khi flatten, các nhãn có trong tập dữ liệu.
* Hiển thị 10 ảnh ngẫu nhiên cùng với nhãn tương ứng để có cái nhìn trực quan về dữ liệu.
  + 1. Chia dữ liệu:



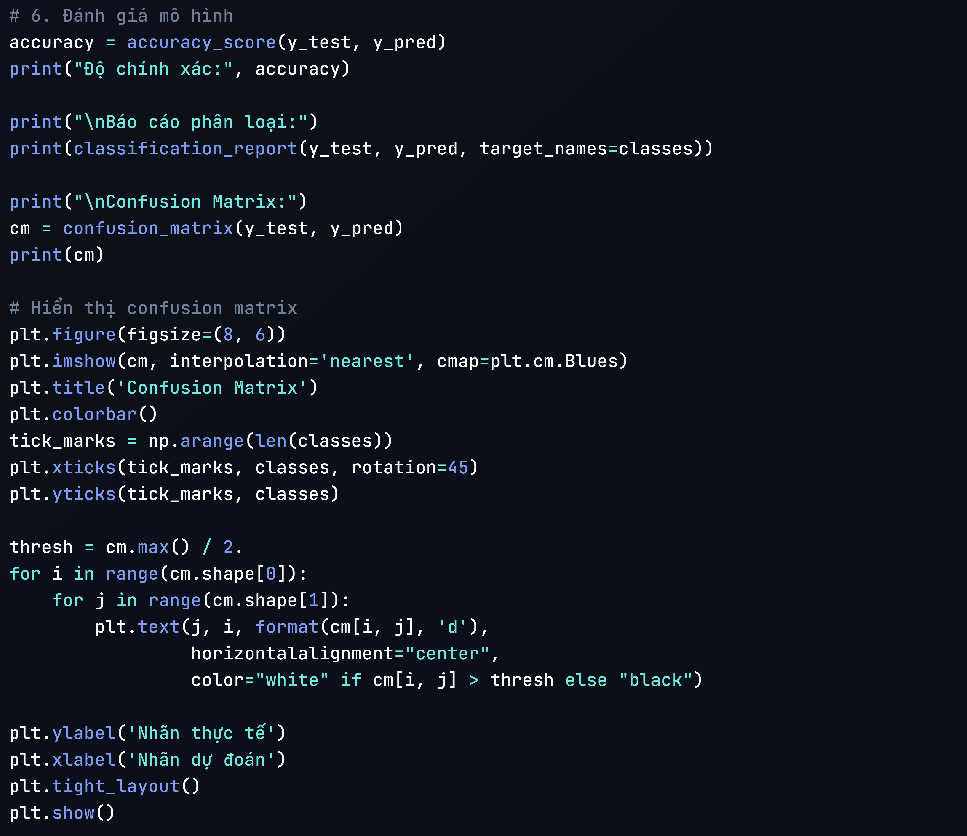
* train\_test\_split(data, labels, test\_size=0.2, random\_state=42): Chia dữ liệu thành 2 phần:
  + **Tập huấn luyện (training set):** Dùng để huấn luyện mô hình (80% dữ liệu).
  + **Tập kiểm tra (test set):** Dùng để đánh giá mô hình sau khi huấn luyện (20% dữ liệu).
  + X\_train, y\_train: Dữ liệu và nhãn cho tập huấn luyện.
  + X\_test, y\_test: Dữ liệu và nhãn cho tập kiểm tra.
  + random\_state=42: Đảm bảo việc chia dữ liệu là nhất quán (mỗi lần chạy code sẽ cho kết quả chia giống nhau).
    1. Huấn luyện mô hình:



* model = GaussianNB(): Tạo một đối tượng mô hình GaussianNB.
* model.fit(X\_train, y\_train): Huấn luyện mô hình sử dụng dữ liệu và nhãn từ tập huấn luyện. Mô hình sẽ "học" mối quan hệ giữa các đặc trưng (pixel của ảnh) và nhãn.
  + 1. Dự đoán:



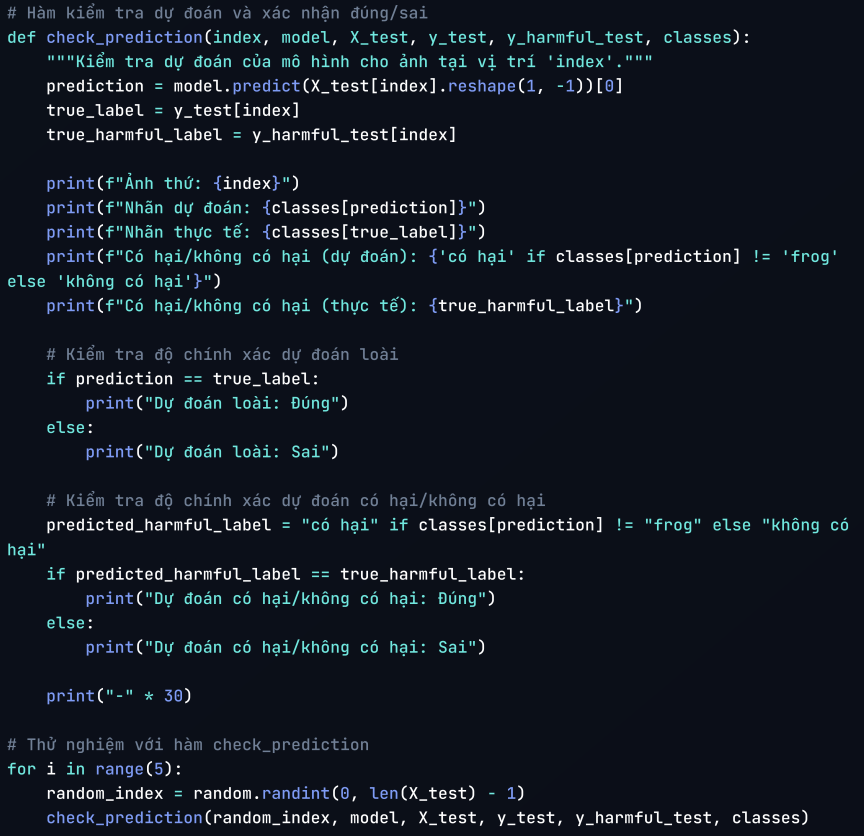
* y\_pred = model.predict(X\_test): Sử dụng mô hình đã huấn luyện để dự đoán nhãn cho các ảnh trong tập kiểm tra. Kết quả dự đoán được lưu trong y\_pred.
  + 1. Đánh giá mô hình:



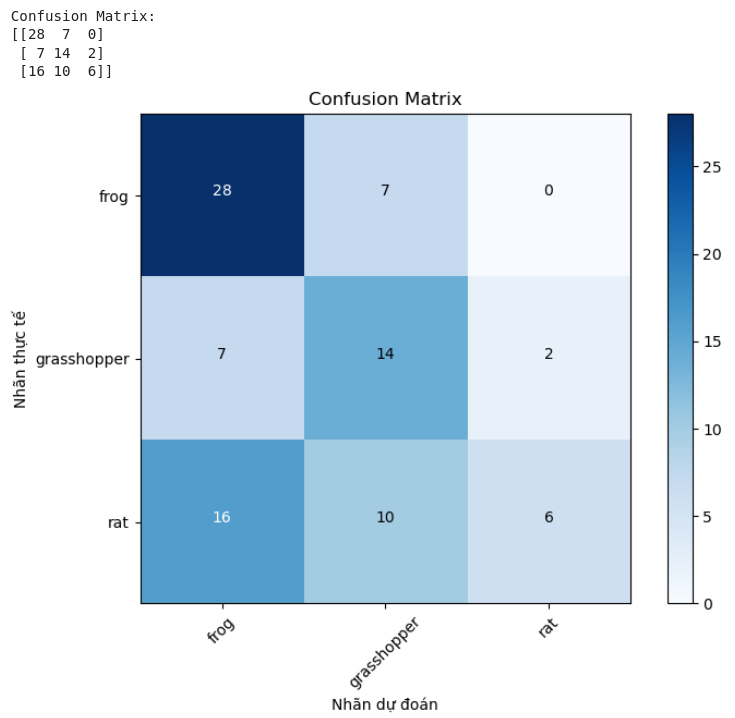
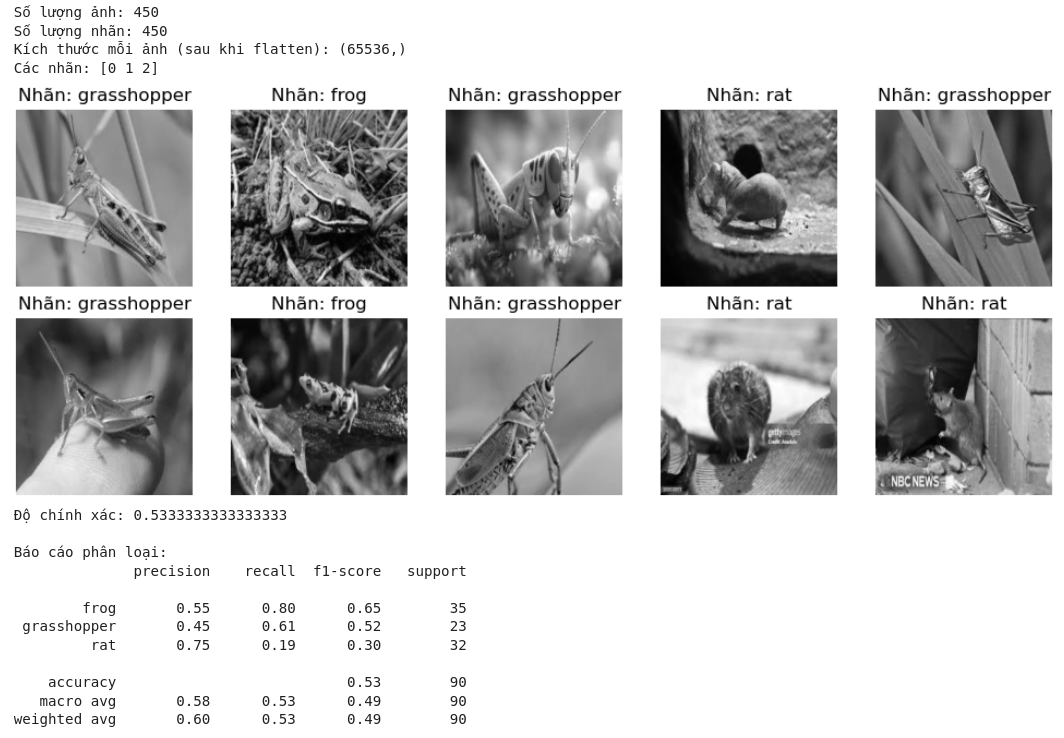
* **accuracy\_score(y\_test, y\_pred):** Tính toán độ chính xác tổng thể của mô hình (tỷ lệ phần trăm số ảnh được dự đoán đúng).
* **classification\_report(y\_test, y\_pred, target\_names=classes):** In ra báo cáo phân loại chi tiết hơn, bao gồm:
  + **Precision:** Độ chính xác cho từng lớp (trong số các ảnh được dự đoán là lớp X, có bao nhiêu ảnh thực sự là lớp X).
  + **Recall:** Độ phủ cho từng lớp (trong số các ảnh thực sự là lớp X, có bao nhiêu ảnh được mô hình dự đoán đúng là lớp X).
  + **F1-score:** Trung bình điều hòa của precision và recall.
  + **Support:** Số lượng mẫu thực tế của từng lớp trong tập kiểm tra.
* **confusion\_matrix(y\_test, y\_pred):** Tạo ma trận nhầm lẫn (confusion matrix). Ma trận này cho biết số lượng ảnh thực tế thuộc lớp i nhưng được dự đoán là lớp j.
* Phần code còn lại dùng để hiển thị confusion matrix dưới dạng biểu đồ, giúp dễ dàng quan sát và đánh giá hiệu suất của mô hình.
  + 1. Thêm nhãn "có hại/không có hại" và chia tập dữ liệu



* **harmful\_labels = []:** Tạo một list rỗng để lưu trữ nhãn "có hại/không có hại".
* **for label in labels::** Lặp qua từng nhãn trong labels (0, 1, 2).
  + **if classes[label] == "frog"::** Nếu nhãn tương ứng với lớp "frog".
    - **harmful\_labels.append("không có hại"):** Thêm "không có hại" vào harmful\_labels.
  + **else::** Nếu nhãn không phải là "frog" (là "grasshopper" hoặc "rat").
    - **harmful\_labels.append("có hại"):** Thêm "có hại" vào harmful\_labels.
* **harmful\_labels = np.array(harmful\_labels):** Chuyển harmful\_labels thành mảng NumPy.
* **y\_harmful\_train, y\_harmful\_test = train\_test\_split(harmful\_labels, test\_size=0.2, random\_state=42)**: Chia tập harmful\_labels thành tập train và test theo tỷ lệ 8:2
* **print("Các nhãn có hại/không có hại:", np.unique(harmful\_labels)):** In ra các giá trị duy nhất trong harmful\_labels (sẽ là "có hại" và "không có hại").
  + 1. Hàm kiểm tra dự đoán và sau đó thực hiện dự đoán



* **def check\_prediction(index, model, X\_test, y\_test, y\_harmful\_test, classes)::** Định nghĩa hàm check\_prediction nhận vào các tham số:
  + **index:** Vị trí của ảnh trong tập kiểm tra cần kiểm tra.
  + **model:** Mô hình đã huấn luyện.
  + **X\_test:** Dữ liệu kiểm tra.
  + **y\_test:** Nhãn thực tế của tập kiểm tra.
  + **y\_harmful\_test:** Nhãn "có hại/không có hại" thực tế của tập kiểm tra.
  + **classes:** Danh sách tên các lớp.
* **prediction = model.predict(X\_test[index].reshape(1, -1))[0]:** Lấy ảnh tại vị trí index từ X\_test, reshape thành dạng mà mô hình có thể nhận (1 dòng, -1 tự động tính số cột), đưa vào model.predict để dự đoán nhãn, và lấy giá trị dự đoán đầu tiên [0].
* **true\_label = y\_test[index]:** Lấy nhãn thực tế của ảnh từ y\_test.
* **true\_harmful\_label = y\_harmful\_test[index]:** Lấy nhãn "có hại/không có hại" thực tế của ảnh từ y\_harmful\_test.
* **Các lệnh print:** In ra các thông tin:
  + Ảnh thứ mấy đang được kiểm tra.
  + Nhãn dự đoán (tên lớp).
  + Nhãn thực tế (tên lớp).
  + Nhãn "có hại/không có hại" dự đoán (dựa vào nhãn dự đoán, nếu là "frog" thì "không có hại", ngược lại là "có hại").
  + Nhãn "có hại/không có hại" thực tế.
  + Kết luận dự đoán loài đúng hay sai
  + Kết luận dự đoán có hại/không có hại đúng hay sai
* **print("-" \* 30):** In ra một dòng gạch ngang để phân cách các lần kiểm tra.
* **for i in range(5)::** Lặp 5 lần.
  + **random\_index = random.randint(0, len(X\_test) - 1):** Chọn một index ngẫu nhiên trong tập kiểm tra.
  + **check\_prediction(random\_index, model, X\_test, y\_test, y\_harmful\_test, classes):** Gọi hàm check\_prediction để kiểm tra dự đoán cho ảnh tại random\_index.

1. Kết quả chương trình:

**Kết quả tổng quan:**

* **Độ chính xác (Accuracy):** 0.53 (53%)
* Mô hình dự đoán đúng 48/90 mẫu (53%), 42/90 mẫu (47%) bị phân lớp sai.

**Chi tiết từng lớp (theo classification\_report):**

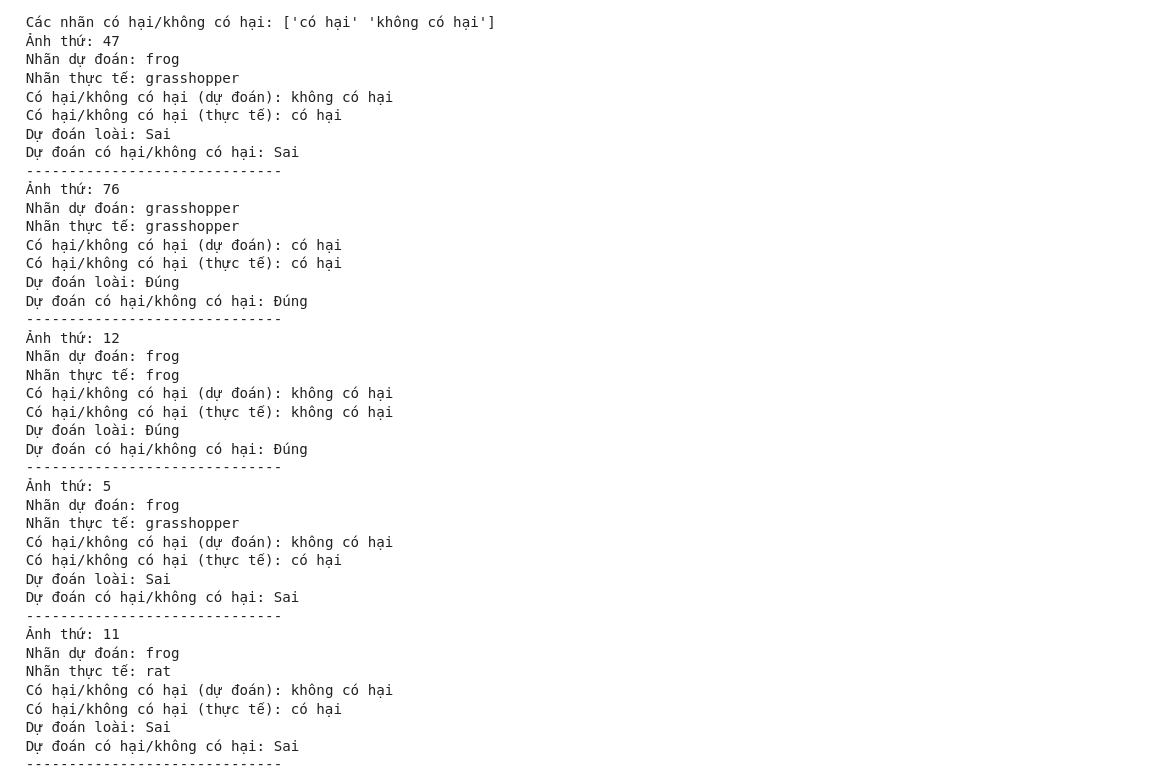
* **Frog:**
  + **Precision:** 0.55 (55%) - Trong số các mẫu được dự đoán là frog, chỉ có 55% là chính xác.
  + **Recall:** 0.80 (80%) - Trong số các mẫu thực sự là frog, mô hình nhận diện đúng 80%.
  + **F1-score:** 0.65 (65%) - Trung bình điều hòa giữa Precision và Recall.
  + **Support:** 35 - Số lượng mẫu frog thực tế trong tập kiểm tra.
* **Grasshopper:**
  + **Precision:** 0.45 (45%)
  + **Recall:** 0.61 (61%)
  + **F1-score:** 0.52 (52%)
  + **Support:** 23
* **Rat:**
  + **Precision:** 0.75 (75%)
  + **Recall:** 0.19 (19%)
  + **F1-score:** 0.30 (30%)
  + **Support:** 32

**Ma trận nhầm lẫn (Confusion Matrix):**

* **Frog:**
  + 28 mẫu được dự đoán đúng vào lớp frog.
  + 7 mẫu bị dự đoán nhầm thành grasshopper.
  + 0 mẫu bị dự đoán nhầm thành rat.
* **Grasshopper:**
  + 7 mẫu bị dự đoán nhầm thành frog.
  + 14 mẫu được dự đoán đúng vào lớp grasshopper.
  + 2 mẫu bị dự đoán nhầm thành rat.
* **Rat:**
  + 16 mẫu bị dự đoán nhầm thành frog.
  + 10 mẫu bị dự đoán nhầm thành grasshopper.
  + 6 mẫu được dự đoán đúng vào lớp rat.

**Kết luận:** Mô hình Naive Bayes cho kết quả Accuracy đạt 53%. Mô hình dự đoán tốt nhất ở lớp frog (F1-score 65%) và kém nhất ở lớp rat (F1-score chỉ 30%). Nhìn chung, mô hình còn nhầm lẫn khá nhiều giữa các lớp, đặc biệt rat thường bị nhận nhầm thành frog hoặc grasshopper

**Kết quả kiểm tra thực tế:**

****

Đánh giá chung:

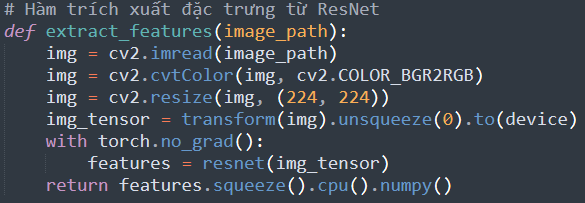
* Độ chính xác dự đoán có hại/không có hại: 2/5 (40%)
* Mô hình có xu hướng dự đoán các loài vật là frog (3/5 trường hợp), dẫn đến dự đoán sai về mức độ có hại/không có hại.
* Cần cải thiện mô hình để phân biệt tốt hơn giữa các loài, đặc biệt là giữa frog với grasshopper và rat.

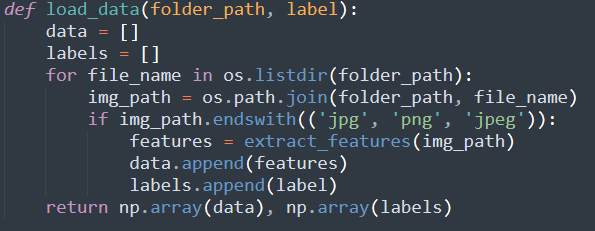
## Xây dựng mô hình K-Nearest Neighbors

1. Ngôn ngữ: Python.
2. Trình soạn thảo: Sublime Text
3. Các thư viện sử dụng:

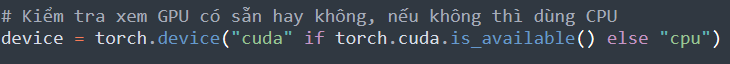
* Numpy: Xử lý mảng số liệu hiệu quả.
* Cv2 (OpenCV): Dùng để xử lý hình ảnh (đọc, resize, chuyển màu).
* os: Xử lý đường dẫn và file hệ thống.
* sklearn: Sử dụng các mô hình học máy (KNN) và các công cụ khác như chia dữ liệu, tính độ chính xác.
* torch và torchvision: Thư viện học sâu và các mô hình pretrained để trích xuất đặc trưng từ ảnh.
* time: Đo thời gian thực hiện của các phần trong chương trình.

1. Các bước xây dựng:

* Hàm trích xuất đặc trưng từ Resnet  
  
  + Ảnh đầu vào được chuẩn hóa và resize về kích thước phù hợp với mô hình ResNet (224x224).
  + Resnet được sử dụng để trích xuất đặc trưng từ ảnh (bỏ qua lớp cuối, chỉ lấy đầu ra của lớp feature).
* Hàm load dữ liệu và trích xuất đặc trưng:

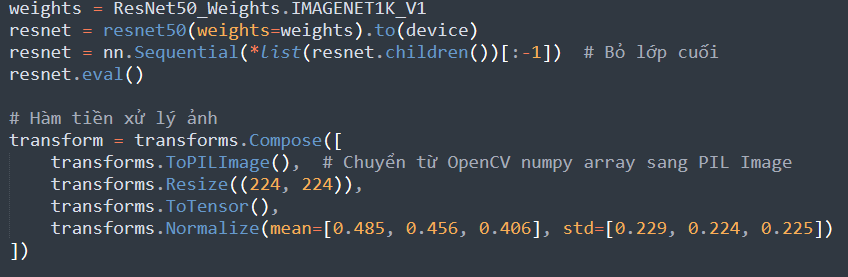


* + Hàm này duyệt qua thư mục chứa ảnh, trích xuất đặc trưng cho từng ảnh và gán nhãn phù hợp (0: Ếch, 1: Châu chấu, 2: Chuột).
* Hàm dự đoán:   
  
  + Hàm này dùng để dự đoán loài của một ảnh và chỉ ra mức độ nguy hại.
* Thiết lập thiết bị



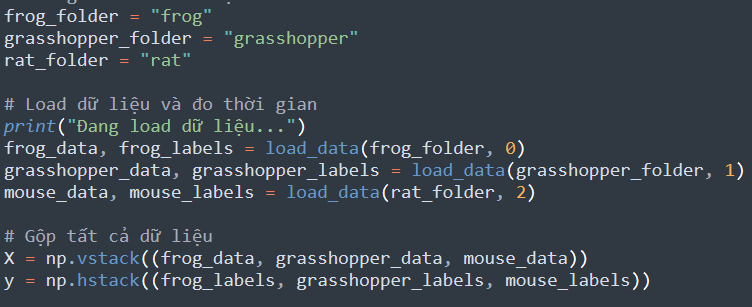
* + Chương trình tự động kiểm tra GPU có sẵn hay không. Nếu có, dùng GPU để tăng tốc độ xử lý.

* Tải mô hình Resnet và Pipeline tiền xử lý dữ liệu

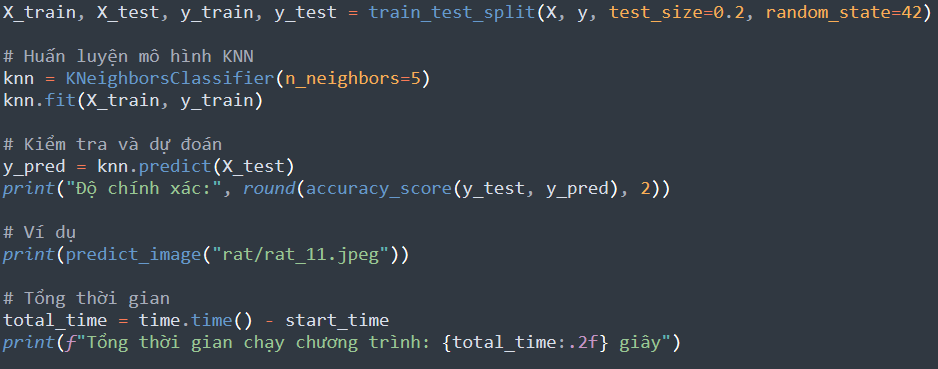


* + Resnet50: Mô hình ResNet50 pretrained trên ImageNet, được sử dụng để trích xuất đặc trưng từ ảnh.
  + Transforms: Pipeline tiền xử lý ảnh, bao gồm resize, chuyển đổi sang tensor và chuẩn hóa theo chuẩn ImageNet.

* Load dữ liệu

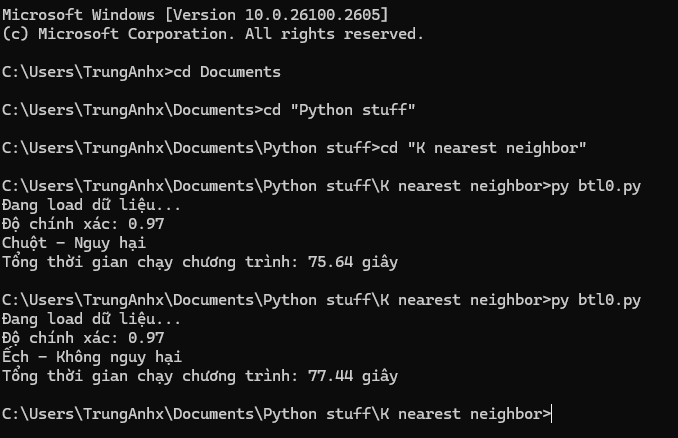


* + Chương trình duyệt qua từng thư mục ảnh của 3 loài, trích xuất đặc trưng và gán nhãn.
  + Dữ liệu của tất cả các loài được gộp lại thành x(đặc trưng) và y (nhãn).

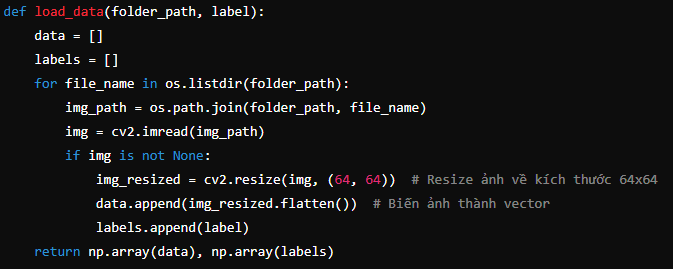
* Chia dữ liệu, huấn luyện mô hình và tính độ chuẩn xác  
  
  + Dữ liệu được chia thành tập huấn luyện (80%) và tập kiểm tra (20%).
  + Mô hình KNN được huấn luyện trên tập huấn luyện (ở đây tôi chọn k = 5 là số lượng láng giềng).
  + Mô hình dự đoán trên tập kiểm tra (X\_test) và so sánh với nhãn thật sự (y\_test) để tính độ chính xác.
  + Thời gian chạy toàn bộ chương trình từ lúc bắt đầu đến lúc kết thúc được đo và in ra.

1. Kết quả K-Nearest Neighbors:

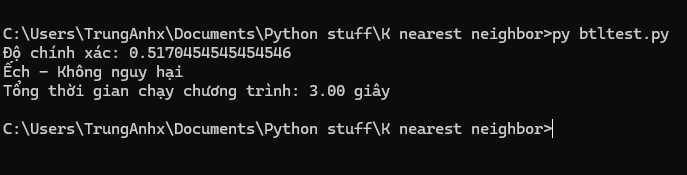
* Kết quả chạy code:



* So sánh kết quả thuật toán:
* Nếu không sử dụng Resnet để làm trích xuất đặc trưng và tiền xử lý dữ liệu, thay vào đó là sử dụng các giá trị pixel thô.



* Ở đây hàm load data pixel thô để biến thành vector.
* Kết quả chính xác của thuật toán bị ảnh hưởng khá nhiều trong khi thời gian chạy lại nhanh hơn.



# **KẾT LUẬN**

Nhu cầu đảm bảo an ninh và tối ưu hóa hiệu quả sản xuất trong nông nghiệp ngày càng cao đã thúc đẩy việc ứng dụng các công nghệ tiên tiến vào giám sát và bảo vệ khu vực canh tác. Trong đó, hệ thống phát hiện và cảnh báo sớm động vật xâm nhập, dựa trên nền tảng công nghệ phân lớp dữ liệu, đang nổi lên như một giải pháp đầy tiềm năng. Việc kết hợp giữa xử lý ảnh, học máy và các thuật toán phân lớp mạnh mẽ cho phép tự động hóa quá trình giám sát, nhận diện chính xác loài động vật xâm nhập và đưa ra cảnh báo kịp thời, góp phần giảm thiểu thiệt hại và nâng cao năng suất trong nông nghiệp.

Nhằm ứng dụng kiến thức đã học vào thực tiễn và trau dồi kỹ năng trong lĩnh vực khai phá dữ liệu, nhóm chúng em đã được phân công thực hiện đề tài: "Xây dựng hệ thống phát hiện và cảnh báo động vật xâm nhập khu nông nghiệp sử dụng các thuật toán Random Forest, K-Nearest Neighbors và Naive Bayes". Nhận thấy đây là một đề tài thiết thực, có tính ứng dụng cao, và phù hợp với định hướng học tập về xử lý ảnh, nhóm chúng em đã bắt tay vào thực hiện. Sau một quá trình nghiêm túc nghiên cứu và triển khai, chúng em đã đạt được những kết quả nhất định như sau:

* Chúng em cũng đã nghiên cứu sâu về nguyên lý hoạt động, ưu nhược điểm của các thuật toán Random Forest, K-Nearest Neighbors và Naive Bayes. Trên cơ sở đó, nhóm đã bước đầu xây dựng được một hệ thống có khả năng phát hiện, phân loại và cảnh báo về sự hiện diện, loài cũng như tác động của chúng khi xâm nhập khu nông nghiệp.
* Không chỉ dừng lại ở lý thuyết, chúng em đã thực hành thu thập và xử lý dữ liệu ảnh, sử dụng các công cụ và thư viện như: OpenCV, Scikit-learn, TensorFlow, Keras để tiền xử lý dữ liệu, trích xuất đặc trưng từ ảnh sử dụng mô hình ResNet pre-trained, và xây dựng các mô hình phân lớp Random Forest, K-Nearest Neighbors, Naive Bayes trên Python.
* Qua quá trình này, chúng em đã có cơ hội so sánh và đánh giá hiệu suất của ba mô hình phân lớp, từ đó rút ra những kinh nghiệm quý báu về ưu điểm, nhược điểm của từng mô hình trong bài toán thực tế này, và tìm ra phương pháp kết hợp các mô hình để đạt hiệu quả tối ưu.

Trong quá trình thực hiện, dù đã nỗ lực không ngừng, nhóm chúng em nhận thấy rằng vẫn còn đó những hạn chế và điểm chưa hoàn hảo. Chúng em hi vọng sẽ nhận được những lời góp ý quý báu từ các thầy cô, để từ đó, chúng em có thể học hỏi, phát triển và hoàn thiện hơn nữa trên con đường khám phá tri thức này.

# **TÀI LIỆU THAM KHẢO**

* Nguồn ảnh: [Bing Images](https://www.bing.com/images/), <https://images.google.com/>
* Nguồn tài liệu lý thuyết:
  + Random Forest**:**

<https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/>

<https://scikit-learn.org/stable/modules/ensemble.html#random-forests>

* + K-Nearest Neighbors:

<https://www.geeksforgeeks.org/k-nearest-neighbours/>

<https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors>

* + Naive Bayes Classification:

<https://www.geeksforgeeks.org/naive-bayes-classifiers/>

<https://scikit-learn.org/stable/modules/naive_bayes.html>

* Nguồn tài liệu về thư viện:
* Scikit-learn: <https://scikit-learn.org/stable/>
* Pytorch: <https://pytorch.org/>
* OpenCV: <https://opencv.org/>
* Tài liệu TensorFlow cho ResNet:

[Module: tf.keras.applications.resnet  |  TensorFlow v2.16.1](https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet)