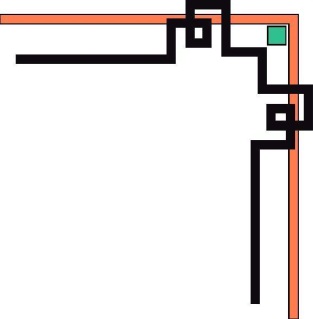
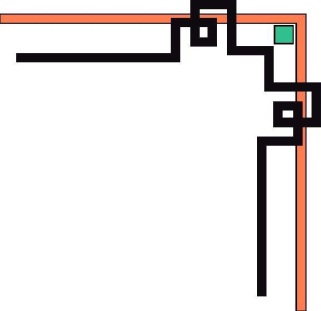
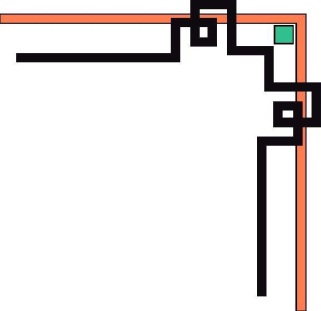
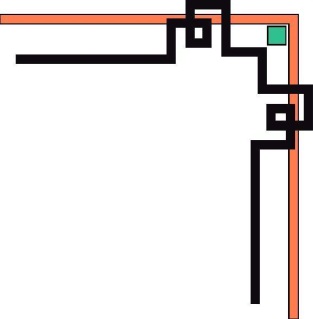
**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA ĐIỆN – ĐIỆN TỬ**

**🖎🕮✍**





**BÁO CÁO BÀI TẬP LỚN XỬ LÝ ẢNH**

**ĐỀ TÀI: BỘ PHÂN LỚP NAÏVE BAYES**

**GVHD: Võ Tuấn Kiệt**

Nhóm 02 – A02

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** |
| **1** | **Nguyễn Gia Khiêm** | **1810236** |
| **2** | **Thái Quang Nguyên** | **1813294** |
| **3** | **Võ Quốc Huy** | **1711571** |
| **4** | **Lê Phúc Công Thành** | **1713150** |

**Tp.HCM, tháng 11 năm 2020**

**MỤC LỤC**

[**I. Giới thiệu về bộ phân lớp Navie Bayes** 1](#_Toc56943305)

[**1. Định nghĩa** 1](#_Toc56943306)

[**2. Các mô hình thuật toán Navie Bayes Classification** 1](#_Toc56943307)

[a) Mô hình Gaussian 1](#_Toc56943308)

[b) Mô hình Bernoulli 1](#_Toc56943309)

[c. Mô hình Multinomial 2](#_Toc56943310)

[**II. Thực hiện code nhận dạng hình ảnh** 5](#_Toc56943312)

[**1.** **Cấu trúc thư mục và thư viện:** 5](#_Toc56943313)

[**2.** **Chuẩn bị tập dữ liệu:** 6](#_Toc56943314)

[**3. Rút trích đặc trưng:** 11](#_Toc56943315)

[**4. Train và Test các bộ phân lớp** 12](#_Toc56943316)

[a. Traning Classifier 12](#_Toc56943317)

[b. Traning & Testing Classifier 14](#_Toc56943318)

[**III. Kết quả và nhận xét:** 18](#_Toc56943319)

[**1.** **Kết quả thu được khi chạy code test:** 18](#_Toc56943320)

[**2.** **So sánh, nhận xét:** 19](#_Toc56943321)

[**III. Tài liệu tham khảo** 20](#_Toc56943322)

# **I. Giới thiệu về bộ phân lớp Navie Bayes**

## **1. Định nghĩa**

Navie Bayes Classification (NBC) là một thuật toán phân loại dựa trên tính toán xác suất áp dụng định lý Bayes. Thuật toán này thuộc nhóm Supervised Learning (Học có giám sát).

Theo định lý Bayes, ta có công thức tính xác suất ngẫu nhiên của sự kiện y khi biết x như sau:



Trên thực tế thì ít khi tìm được dữ liệu làm các thành phần là hoàn toàn độc lập với nhau. Tuy nhiên, giả thiết này giúp cách tính toán trở nên đơn giản, training data nhanh, đem lại hiệu quả bất ngờ và các lớp bài toán nhất định.

Cách xác định các thành phần (class) của dữ liệu dựa trên giả thiết này có tên là Navie Bayes Classification.

## **2. Các mô hình thuật toán Navie Bayes Classification**

Có 3 mô hình thuật toán Naive Bayes thường sử dụng là: Gaussian, Bernoulli và Multinomial.

### **a) Mô hình Gaussian**

Được sử dụng cho đối tượng có dữ liệu là biến liên tục:



Với 

### **b) Mô hình Bernoulli**

Ở mô hình này, các feature vector là các giá trị nhị phân 0, 1. Trong đó 1 thể hiện từ đó xuất hiện trong văn bản, 0 được thể hiện từ đó không xuất hiện trong văn bản.

#### Công thức:

#### 

Hoặc theo công thức: 

Ví dụ: Cho bảng sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Confident | Studied | Sick | Result |
| Yes | No | No | Fail |
| No | No | Yes | Pass |
| No | Yes | Yes | Fail |
| Yes | Yes | No | Pass |
| Yes | Yes | Yes | Pass |

Xét một tập dữ liệu X mà ta muốn phân loại: Confident = Yes, Studied = Yes và Sick = No.

+ Trước tiên cần tính xác suất của lớp: P(Pass) = 3/5 và P(Fail) = 2/5.

+ Tính xác suất từng tính năng:

Bây giờ chúng ta tính toán

**Vì , trường hợp X có kết quả là Pass**

### **c. Mô hình Multinomial**

Mô hình này chủ yếu được dùng cho việc phân loại văn bảng dựa trên ý tưởng BoW (Bag of Words). Mỗi văn bản được thể hiện thành dạng bag of words, hiểu nôm na là thể hiện xem có bao nhiêu từ xuất hiện và tần suất xuất hiện trong văn bản, nhưng bỏ qua thứ tự các từ.

Ở mô hình này, các feature vector là các giá trị số tự nhiên mà giá trị thể hiện số lần từ đó xuất hiện trong văn bản. Ta tính xác suất từ xuất hiện trong văn bản  như sau:

****

Trong đó: Ni  là tổng số lần xi xuất hiện trong văn bản.

Nc là tổng số lần từ của tất cả các từ x1, …xn xuất hiện trong văn bản

##### Laplace Smoothing

Công thức trên có hạn chế là khi xi  không xuất hiện lần nào, ta sẽ có Ni = 0, điều này làm cho P(xi | c) = 0.

Để khắc phục vấn đề này, người ta sử dụng kỹ thuật gọi là Laplace Smoothing bằng cách cộng thêm vào cả tử và mẫu giá trị luôn khác 0.



Trong đó: α thường là số dương bằng 1

dα được cộng vào để đảm bảo 

|  |  |  |  |
| --- | --- | --- | --- |
|  | DOC | WORD | CLASS |
| Training | 1 | Chinese Bejing Chinese | c |
|  | 2 | Chinese Chinese Shanghai | c |
|  | 3 | Chinese Macao | c |
|  | 4 | Tokyo Japan Chinese | j |
| Test | 5 | Chinese Chinese Chinese Tokyo Japan | ? |

Ta có: ,

+ (5 là số từ Chinese trong cụm class c, 8 là số lượng từ trong phân cụm class c, 6 là số term khác nhau cũng trong c)

Cuối cùng chúng ta tính ra được xác xuất để chọn class:

Vậy là chúng ta chọn loại c cho **Doc5**

* **Nhận xét**

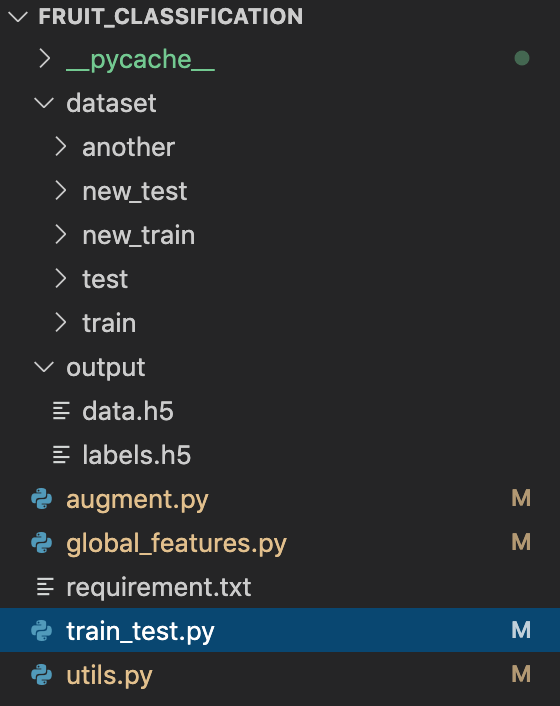
Naive Bayes Classifiers (NBC) là phương pháp cổ điển nhưng vẫn rất hữu dụng với các bài toán nhất định như phân loại văn bản, email…

NBC với công thức tính toán đơn giản nên dễ cài đặt (hiện nay nếu dùng thư viện sklearn thì chỉ cần gọi vài dòng lệnh như mình làm bên trên), thời gian training và test nhanh, phù hợp với bài toán data lớn.

Cần chú ý sử dụng Smoothing để tránh lỗi xác suất tổng được bằng 0 khi xác suất của một feature thành phần bằng 0.

# **II. Thực hiện code nhận dạng hình ảnh**

1. **Cấu trúc thư mục và thư viện:**



Ta có cấu trúc thư mục như hình bên, trong đó folder *dataset* chứa hai folder con *new\_test* và *new\_train* chứa tập file hình ảnh để test và train. Folder *train* là folder chứa train data được tải xuống từ source chưa qua hiệu chỉnh. File ảnh được lưu theo cú pháp:

***“image (n).jpg”*** trong đó *n >= 1* để tiện truy xuất.

Folder output được dùng để lưu những đặc trưng (file data.h5) và các nhãn tương ứng (tên loại hoa quả) của mỗi hình (labels.h5).

Có nhiều định dạng để lưu những đặc trưng này như .*xlsx, .xml,…* Tuy nhiên với định dạng *.h5*, khi tập dữ liệu càng lớn thì dung lượng của file sẽ càng nhẹ hơn so với các định dạng còn lại.

**Các thư viện cần có: (chèn link hướng dẫn & cách cài đặt)**

- *opencv-contrib-lib (OpenCV)*: thư viện cần có để thực hiện những tác vụ với hình ảnh

- *matplotlib*: hiển thị plot dạng ma trận

- *h5py*: dùng để xử lý file có định dạng .h5

- *scikit-learn:* bộ thư viện dùng để thực hiện Machine Learning

- *mahotas*: sử dụng kết hợp với đặc trưng hoa văn Haralick Texture

1. **Chuẩn bị tập dữ liệu:**

Xuất phát từ tính đa dạng về hình dáng, màu sắc dẫn đến sự đa dạng về các đặc trưng, nhóm chúng em lựa chọn tập [dữ liệu hoa quả của Keggel](https://www.kaggle.com/moltean/fruits) để hiện thực hoá tác dụng phân loại của bộ phân lớp Naïve Bayes.

***Ưu điểm của bộ dataset hoa quả Kegel:***

* Được tách nền sẵn dẫn đến ít bị nhiễu khi trích xuất đặc trưng.
* Bao gồm phong phú các loại hoa quả.
* Hình ảnh được resized về cùng một resolution 100x100 pixels.

***Nhược điểm của bộ dataset:***

* Các hình ảnh trong cùng một loại hoa quả là chụp từng góc của vật thể, với các đặc trưng gần như giống nhau. Dẫn đến việc máy không học được nhiều, sau cùng sẽ dẫn đến việc dự đoán sai lệch.

Do đó, để có thể tạo ra một bộ phân lớp tối ưu, ta cần đa dạng hoá bộ training data. Nhóm thực hiện theo những cách sau đây:

Đầu tiên, chọn ra trong một loại hoa quả những hình khác nhau nhất.

Tiếp theo, sử dụng kết hợp những hàm sau đây để đa dạng hoá train data (các hàm này được định nghĩa trong file utils.py (*insert hyperlink*) đi kèm):

* Hàm random\_shadow(image) với tác dụng thay đổi những vùng tối của bức ảnh:

def random\_shadow(image):

# ----- Generates and adds random shadow

hsvImg = cv2.cvtColor(image, cv2.COLOR\_RGB2HLS)

hsvImg[..., 2] = hsvImg[..., 2] \* random.uniform(0.3, 0.6)

return cv2.cvtColor(hsvImg, cv2.COLOR\_HLS2RGB)

* Hàm random\_brightness(image) có tác dụng tăng giảm độ sáng của toàn bức ảnh:

def random\_brightness(image):

"""

Randomly adjust brightness of the image.

"""

# -----Converting image to LAB Color model-----------------------------------

lab = cv2.cvtColor(image, cv2.COLOR\_BGR2LAB)

# -----Splitting the LAB image to different channels-------------------------

l, a, b = cv2.split(lab)

# -----Applying CLAHE to L-channel-------------------------------------------

clahe = cv2.createCLAHE(clipLimit=random.uniform(1, 5), tileGridSize=(8, 8))

cl = clahe.apply(l)

# -----Merge the CLAHE enhanced L-channel with the a and b channel-----------

limg = cv2.merge((cl, a, b))

# -----Converting image from LAB Color model to RGB model--------------------

final = cv2.cvtColor(limg, cv2.COLOR\_LAB2BGR)

return final

* Hàm rotate\_image(image, angle) có tác dụng xoay bức ảnh tới một góc tuỳ ý:

# input là img và góc quay 0-360

def rotate\_image(image, angle):

image\_center = tuple(np.array(image.shape[1::-1]) / 2)

rot\_mat = cv2.getRotationMatrix2D(image\_center, angle, 1.0)

result = cv2.warpAffine(image, rot\_mat, image.shape[1::-1], flags=cv2.INTER\_LINEAR)

return result

* Hàm flip\_image(image, code) có tác dụng lật bức ảnh theo một hướng nào đó:

def flip\_image(image, code):

image = cv2.flip(image, code)

return image

* Hàm contras\_image(image, alpha, beta) với tác dụng điều chỉnh độ tương phản:

def contras\_image(image, alpha, beta):

new\_image = np.zeros(image.shape, image.dtype)

for y in range(image.shape[0]):

for x in range(image.shape[1]):

for c in range(image.shape[2]):

new\_image[y, x, c] = np.clip(alpha \* image[y, x, c] + beta, 0, 255)

return new\_image

* Và cuối cùng, vì khi xoay bức ảnh sẽ xuất hiện màu đen ở background trong khi nền của bức ảnh đang là trắng. Để background cùng màu trắng, ta sử dụng bitwise:

def foo(image):

blank = np.zeros((100, 100, 3), dtype='uint8',)

blank[0:100,0:100] = 255, 255, 255

# Create a ROI

rows, cols, channels = image.shape

roi = blank[0:rows, 0:cols]

# Create a mask of fruit image and create its inverse mask also

img2gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

ret, mask = cv2.threshold(img2gray, 1, 255, cv2.THRESH\_BINARY)

mask\_inv = cv2.bitwise\_not(mask)

# Black-out the area of fruit image in ROI

img1\_bg = cv2.bitwise\_and(roi, roi, mask=mask\_inv)

# Take only region of image from fruit image

img2\_fg = cv2.bitwise\_and(image, image, mask=mask)

result\_image = cv2.add(img1\_bg, img2\_fg)

return result\_image

Bây giờ, ta đã có đủ công cụ để làm đầy cho train data của mình, sau đây là thuật toán đọc, tạo thêm hình ảnh và lưu vào folder mới *new\_train* (code được lưu trong file augment.py). File ảnh được lưu theo cú pháp: ***“image (n).jpg”*** trong đó *n >= 1*

import utils

import numpy as np

from cv2 import cv2

import random

import os

image\_path = "dataset/train"

new\_image\_path = "dataset/new\_train"

image\_labels = os.listdir(image\_path)

image\_labels = ['apple','avocado','cauliflower','corn','lime']

print(image\_labels)

for name in image\_labels:

dir = os.path.join(image\_path, name)

new\_dir = os.path.join(new\_image\_path, name)

for x in range(1, 11):

file = dir + "/" + "image (" + str(x) + ").jpg"

count = 1

image = cv2.imread(file)

for i in range(1, 8 + 1): #8

new\_image = utils.foo(utils.rotate\_image(image, random.uniform(0, 360)))

new\_file = new\_dir + "/" + str(count + 40 \* (x - 1)) + ".jpg"

cv2.imwrite(new\_file, new\_image)

count += 1

for i in range(1, 7 + 1): #7

new\_image = utils.contras\_image(image, random.uniform(80, 200) / 100, random.uniform(0, 50) / 10)

new\_file = new\_dir + "/" + str(count + 40 \* (x - 1)) + ".jpg"

cv2.imwrite(new\_file, new\_image)

count += 1

for i in range(1, 7 + 1): #7

new\_image = utils.random\_shadow(image)

new\_file = new\_dir + "/" + str(count + 40 \* (x - 1)) + ".jpg"

cv2.imwrite(new\_file, new\_image)

count += 1

for i in range(-1, 2): #3

new\_image = utils.flip\_image(image, i)

new\_file = new\_dir + "/" + str(count + 40 \* (x - 1)) + ".jpg"

cv2.imwrite(new\_file, new\_image)

count += 1

for i in range(1, 8 + 1): #8

\_image = utils.foo(utils.rotate\_image(image, random.uniform(0, 360)))

new\_image = utils.contras\_image(\_image, random.uniform(80, 200) / 100, random.uniform(0, 50) / 10)

new\_file = new\_dir + "/" + str(count + 40 \* (x - 1)) + ".jpg"

cv2.imwrite(new\_file, new\_image)

count += 1

for i in range(1, 7 + 1): #7

\_image = utils.foo(utils.rotate\_image(image, random.uniform(0, 360)))

new\_image = utils.random\_shadow(\_image)

new\_file = new\_dir + "/" + str(count + 40 \* (x - 1)) + ".jpg"

cv2.imwrite(new\_file, new\_image)

count += 1

Ở đây, nhóm chọn 5 loại hoa quả là apple, avocado, cauliflower, corn, lime để cho ra được mỗi loại 320 bức ảnh .jpg, từng loại được lưu vào folder cùng tên, tất cả là folder con của *new\_train*.

## **3. Rút trích đặc trưng:**

Nhóm chúng em sử dụng 3 loại đặc trưng sau:

* Đặc trưng về màu sắc: [Color Histogram](https://en.wikipedia.org/wiki/Color_histogram)
* Đặc trưng về hình dáng: [Hu Moments](https://en.wikipedia.org/wiki/Image_moment)
* Đặc trưng về hoa văn: [Haralick Texture](http://haralick.org/journals/TexturalFeatures.pdf)

Các hàm dùng để trích xuất các đặc trưng của ảnh:

* **Color Histogram**

Để trích xuất đặc trưng màu theo Color Histogram, ta dùng hàm cv2.calcHist() của OpenCV. Với các đối số là image, channels, mask, histSize (bins) và khoảng giá trị cho mỗi channel (thường từ 0-256). Từ đó ta normalize histogram nhờ vào hàm normalize() của OpenCV, trả về một ma trận được làm phẳng bằng hàm flatten().

# feature-descriptor-3: Color Histogram

def fd\_histogram(image, mask=None):

# convert the image to HSV color-space

image = cv2.cvtColor(image, cv2.COLOR\_BGR2HSV)

# compute the color histogram

hist = cv2.calcHist([image], [0, 1, 2], None, [bins, bins, bins], [0, 256, 0, 256, 0, 256])

# normalize the histogram

cv2.normalize(hist, hist)

# return the histogram

return hist.flatten()

* **Hu Moments**

Dùng hàm cv2.HuMoments() có sẵn của OpenCV. Đối số cho hàm này là moments của hình ảnh đã được làm phẳng. Trước khi làm phẳng hình ảnh, ta chuyển nó về grayscale.

# feature-descriptor-1: Hu Moments

def fd\_hu\_moments(image):#

image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

feature = cv2.HuMoments(cv2.moments(image)).flatten()

return feature

* **Haralick Textures**

Dẫn thư viện mahotas để sử dụng hàm mahotas.feature.haralick(), trước khi sử dụng hàm này ta cần chuyển hình ảnh về grayscale.

# feature-descriptor-2: Haralick Texture

def fd\_haralick(image):#

# convert the image to grayscale

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# compute the haralick texture feature vector

haralick = mahotas.features.haralick(gray).mean(axis=0)

# return the result

return haralick

## **4. Train và Test các bộ phân lớp**

1. **Traning Classifier**

Cùng với các định nghĩa hàm trích xuất đặc trưng ở trên, ta có thể ứng dụng để xuất ra file .h5 chứa đựng các thông số đặc trưng:

# loop over the images in each sub-folder

for x in range(1, images\_per\_class + 1): #images\_per\_class = 320

# get the image file name

file = dir + "/image (" + str(x) + ").jpg"

# read the image and resize it to a fixed-size

image = cv2.imread(file)

image = cv2.resize(image, fixed\_size) #100x100

# Global Feature extraction

fv\_hu\_moments = fd\_hu\_moments(image)

fv\_haralick = fd\_haralick(image)

fv\_histogram = fd\_histogram(image)

# Concatenate global features

global\_feature = np.hstack([fv\_histogram, fv\_haralick, fv\_hu\_moments]) #Stack features

# update the list of labels and feature vectors

labels.append(current\_label)

global\_features.append(global\_feature)

Ở các dòng code trên, ta lần lượt quét tất cả các ảnh của các folder con, để hạn chế sai sót thì gọi them hàm *cv2.resize()* để tất cả các ảnh đều có reso 100x100 pixels. Tiếp đến ta trích đặc trưng của mỗi ảnh bằng 3 hàm *fd\_hu\_moments()*, *fd\_haralick()*, *fd\_histogram()*.

Các giá trị trả về của 3 hàm này sẽ được gộp lại thành một vector *global\_feature* thông qua hàm *numpy.hstack()*.

Mỗi lần quét ảnh, ta thêm vào 2 vector *labels* và *global\_features* nhãn và đặc trưng tương ứng của hình đó.

# encode the target labels

targetNames = np.unique(labels)

le = LabelEncoder()

target = le.fit\_transform(labels)

print("[STATUS] training labels encoded...")

# scale features in the range (0-1)

scaler = MinMaxScaler(feature\_range=(0, 1))

rescaled\_features = scaler.fit\_transform(global\_features)

Việc tiếp theo sẽ là xử lý 2 vector *labels* và *global\_features*.

Khi vector *labels* được append ở trên thì sẽ dẫn đến một loạt các chuỗi sẽ trùng nhau như [‘apple’ ‘apple’ … ’avocado’ ‘avocado’…]. Hàm *numpy.unique()* giúp tìm những labels không lặp lại với nhau. Hàm *LabelEncoder()* và *fit\_transform()* giúp chuyển những chuỗi chữ thành chuỗi số bắt đầu từ 0, ví dụ: [‘apple’ ‘apple’ … ‘lime’ ‘lime’] sẽ trở thành [0 0 … 4 4].

Tương tự, sau khi vector *global\_features* đi qua hàm *fit\_transform()* sẽ được map thành các số từ 0 đến 1, với 0 sẽ tương ứng với số nhỏ nhất trong vector, 1 tương ứng với số lớn nhất.

Tiếp đến, ta tạo 2 file .h5 theo đường dẫn “output/data.h5” và “output/labels.h5” ở chế độ only write ‘w’ và write dữ liệu lên chúng bằng hàm *create\_dataset().*

h5f\_data = h5py.File(h5\_data, 'w')

h5f\_data.create\_dataset('dataset\_1', data=np.array(rescaled\_features))

h5f\_label = h5py.File(h5\_labels, 'w')

h5f\_label.create\_dataset('dataset\_1', data=np.array(target))

h5f\_data.close()

h5f\_label.close()

1. **Traning & Testing Classifier**

Sau khi đã trích xuất được đặc trưng ra thành các vector và lưu vào 2 file .h5, tiếp theo ta sẽ train và test thuật toán của mình. Code phục vụ việc đó được lưu trong file *train\_test.py*.

Đầu tiên mở lại 2 file .h5 và gán 2 vector mới global\_features và global\_labels giá trị của mảng trong 2 file:

# import the feature vector and trained labels

h5f\_data = h5py.File(h5\_data, 'r')

h5f\_label = h5py.File(h5\_labels, 'r')

global\_features\_string = h5f\_data['dataset\_1']

global\_labels\_string = h5f\_label['dataset\_1']

global\_features = np.array(global\_features\_string)

global\_labels = np.array(global\_labels\_string)

h5f\_data.close()

h5f\_label.close()

(trainDataGlobal, testDataGlobal, trainLabelsGlobal, testLabelsGlobal) = train\_test\_split(np.array(global\_features), np.array(global\_labels), test\_size=1/320, random\_state=seed)

clf3 = GaussianNB() #This Naive Bayes

clf3.fit(trainDataGlobal, trainLabelsGlobal)

test\_features = []

test\_results = []

## test all

for testing\_name in test\_labels:

current\_label = testing\_name

# loop over the images in each sub-folder

for x in range(1, images\_per\_class + 1): #image\_per\_class = 80

# get the image file name

file = test\_path + "/" + current\_label + " (" + str(x) + ").jpg"

image = cv2.imread(file)

image = cv2.resize(image, fixed\_size)

fv\_hu\_moments = fd\_hu\_moments(image)

fv\_haralick = fd\_haralick(image)

fv\_histogram = fd\_histogram(image)

test\_results.append(current\_label)

test\_features.append(np.hstack([fv\_histogram, fv\_hu\_moments, fv\_haralick]))

## scale test features

scaler = MinMaxScaler(feature\_range=(0, 1))

rescaled\_features = scaler.fit\_transform(test\_features)

## predict label of test image

le = LabelEncoder()

y\_result = le.fit\_transform(test\_results)

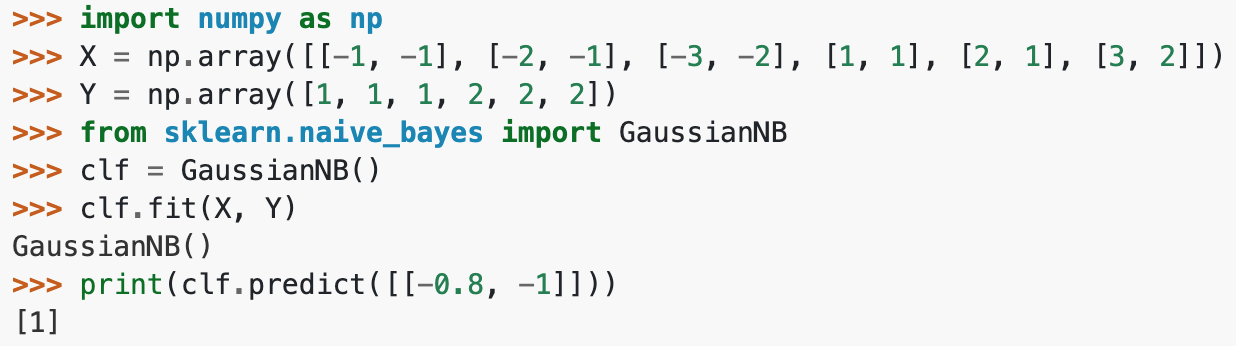
y\_pred = clf3.predict(rescaled\_features)

print("GaussianNB result: ", (y\_pred == y\_result).tolist().count(True)/len(y\_result))

Trước tiên ta lấy một số ảnh để validate từ bộ train data bằng hàm *train\_test\_split()* của sklearn.model\_selection. Gán *clf3* (classifier3) cho GaussianNB và *clf3.fit()* để mỗi giá trị trong vector *trainDataGlobal* sẽ tương ứng với mỗi giá trị trong vector *trainLabelsGlobal*.

Tiếp đến ta sẽ lấy 80 ảnh mỗi loại quả trong thư mục *new\_train* để xem bộ GaussianNB đúng được bao nhiêu phần trăm (validating). So sánh tên của folder (giá trị đúng), với tên mà GaussianNB dự đoán (giá trị dự đoán) sẽ ra được số phần trăm mà GaussianNB dự đoán đúng.

GaussianNB dự đoán bằng cách trích lại đặc trưng của hình ảnh, và thông qua hàm *predict()*, bên dưới là một ví dụ đơn giản:



Tương tự như code validate, ta có thể test và xuất hình ảnh với nhãn dự đoán lên màn hình thông qua các dòng sau:

## show test

for x in range(1, 11):

image\_test\_feature = []

current\_label = train\_labels[random.randint(0, 4)]

index = random.randint(1, 80)

file = test\_path + "/" + current\_label + " (" + str(index) + ").jpg"

print(file)

image = cv2.imread(file)

image = cv2.resize(image, fixed\_size)

fv\_hu\_moments = fd\_hu\_moments(image)

fv\_haralick = fd\_haralick(image)

fv\_histogram = fd\_histogram(image)

test\_features.append(np.hstack([fv\_histogram, fv\_hu\_moments, fv\_haralick]))

image\_rescaled\_features = scaler.fit\_transform(test\_features)

y\_pred = clf3.predict(image\_rescaled\_features)[-1]

print(train\_labels[y\_pred])

cv2.putText(image, train\_labels[y\_pred], (10, 20), cv2.FONT\_HERSHEY\_SIMPLEX, 0.3, (0, 0, 255), 1)

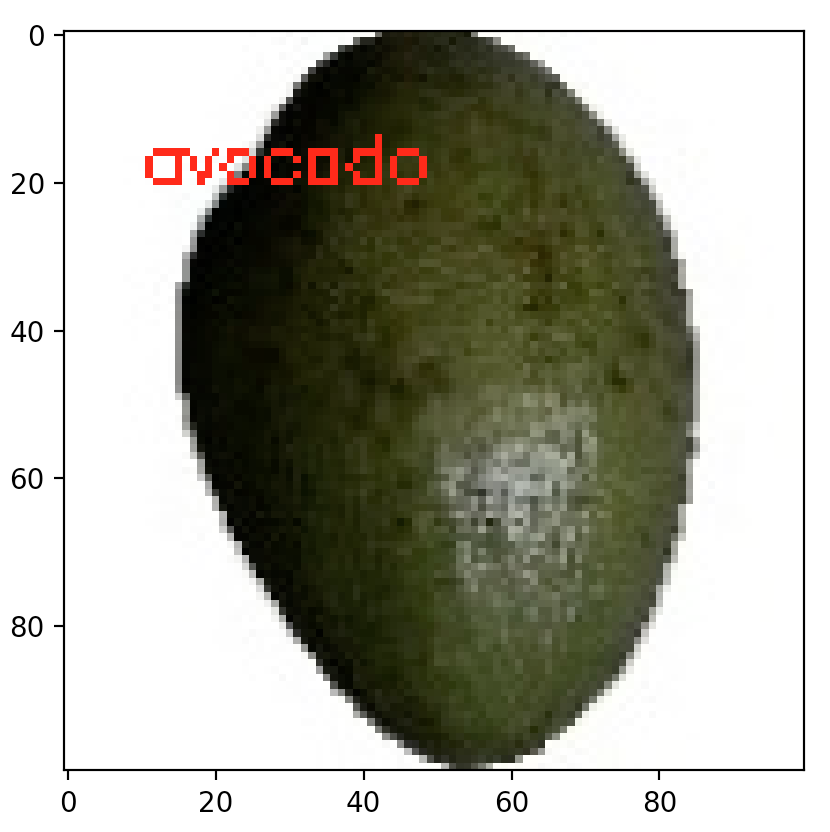
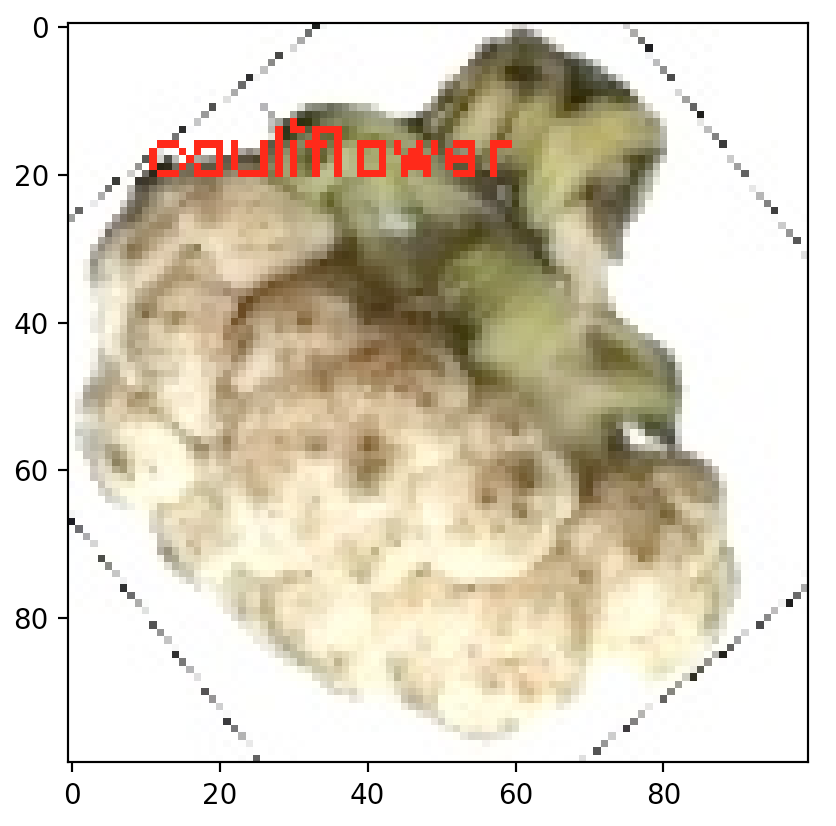
# display the output image

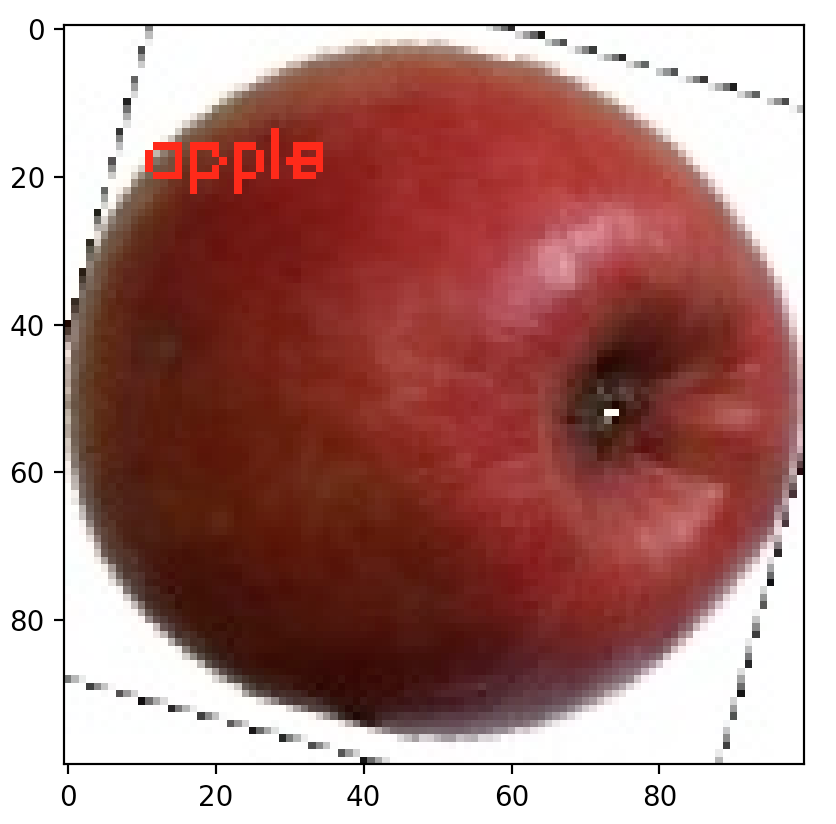
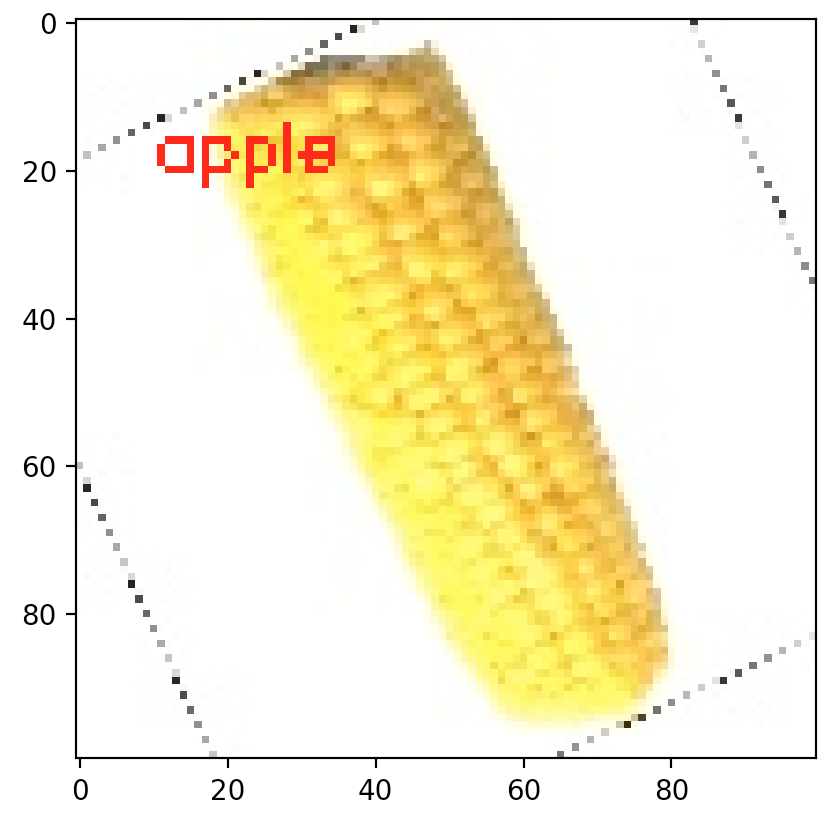
plt.imshow(cv2.cvtColor(image, cv2.COLOR\_BGR2RGB))

plt.show()

# **III. Kết quả và nhận xét:**

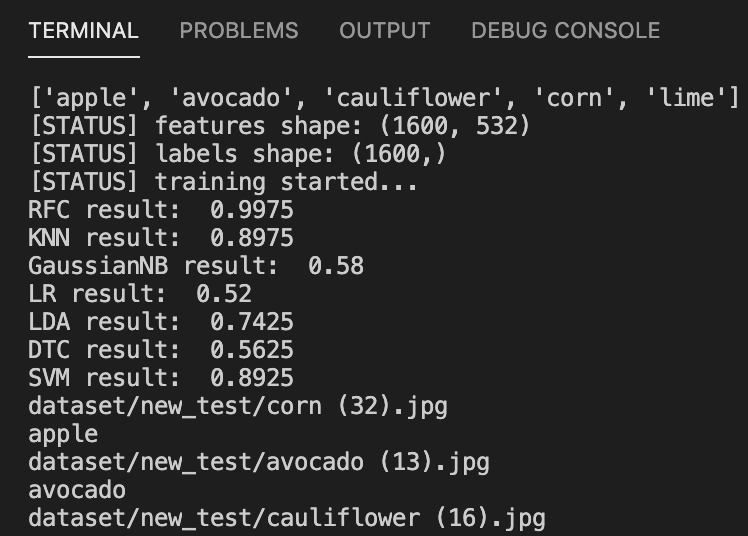
1. **Kết quả thu được khi chạy code test:**

1. **So sánh, nhận xét:**

Thêm các bộ phân lớp khác một cách tương tự, sau khi validate thì ta được độ chính xác của các bộ như sau:



Nhận thấy rằng tỉ lệ đúng của bộ phân lớp GaussianNB chỉ có 58%, so sánh với các bộ khác như RandomForest (99.75%) hay SupportVectorMachine (89.25%) thì không tối ưu bằng.

# **III. Tài liệu tham khảo**

1. <https://dzone.com/articles/naive-bayes-tutorial-naive-bayes-classifier-in-pyt>

2. <https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/>

3. <https://viblo.asia/p/nhan-dang-anh-co-ban-voi-python-bWrZn6mbZxw>

4.<https://gogul.dev/software/image-classification-python?fbclid=IwAR0-Y4OfPrwG3aAqDeNhPDQCIZWgRQA-w7ul1sawcpPKh4uy8GfMwYV5OG0>

5. <https://en.wikipedia.org/wiki/Color_histogram>

6. <https://en.wikipedia.org/wiki/Naive_Bayes_classifier>

7. <https://en.wikipedia.org/wiki/Image_moment>