TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

LÊ ÂU XUÂN DƯƠNG – 19120488

BÁO CÁO HW01

MÔN HỌC: NHẬN DẠNG

CHƯƠNG TRÌNH CHÍNH QUY

GIÁO VIÊN LÝ THUYẾT

Lê Hoàng Thái

GIÁO VIÊN HƯỚNG DẪN THỰC HÀNH

Lê Thanh Phong

Tp. Hồ Chí Minh, ngày 29/04/2023

Mục lục

1. Phân tích bài toán

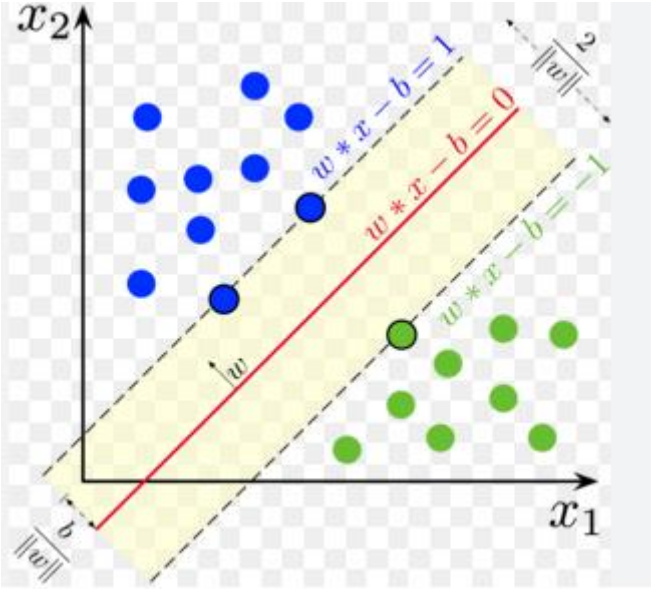
Bài toán này yêu cầu phân lớp các loại bệnh trên cây trồng dựa trên hình ảnh (phân loại dựa trên ảnh). Để giải quyết bài toán này, có thể sử dụng thuật toán SVM hoặc LDA để phân loại. Trong bài này em sử dụng SVM để ứng dụng giải quyết bài toán.

Trong bài toán phân loại dựa trên ảnh, data chúng ta cần xử lý là một loại data có chiều khá lớn (x,x,3). Đối với SVM - là một thuật toán phân lớp được thiết kế để tìm kiếm đường biên (siêu phẳng) tốt nhất để phân tách các điểm dữ liệu của các lớp khác nhau trong không gian đa chiều. Do đó, khi sử dụng SVM để huấn luyện mạng phân lớp, chúng ta có thể đưa hình ảnh từ tập dữ liệu huấn luyện của các loại bệnh khác nhau vào một không gian đa chiều, trong đó đặc trưng của từng hình ảnh sẽ được biểu diễn dưới dạng một vector.

Sau khi các vectơ đặc trưng được biểu diễn, SVM tìm ra đường biên tốt nhất để phân tách các vectơ của các loại bệnh khác nhau trong không gian đa chiều. Khi đã tìm được đường biên tốt nhất, chúng ta có thể sử dụng nó để phân loại các ảnh mới thuộc về một trong các loại bệnh được huấn luyện trước.

a) Tổng quan về SVM

Một máy vectơ hỗ trợ xây dựng một siêu phẳng hoặc một tập hợp các siêu phẳng trong một không gian nhiều chiều hoặc vô hạn chiều, có thể được sử dụng cho phân loại, hồi quy, hoặc các nhiệm vụ khác. Một cách trực giác, để phân loại tốt nhất thì các siêu phẳng nằm ở càng xa các điểm dữ liệu của tất cả các lớp (gọi là hàm lề) càng tốt, vì nói chung lề càng lớn thì sai số tổng quát hóa của thuật toán phân loại càng bé.



Ta có một tập huấn luyện D gồm n điểm có dạng

$$D = \{(x_i, y_i) | x_i \in \mathbb{R}^p, y_i \in \{-1,1\}\}_{i=1}^{n}$$

Với y_i mang giá trị 1 hoặc -1, xác định lớp của điểm x_i. Mỗi x_i là một vector thực p-chiều. Ta cần tìm siêu phẳng có lề lớn nhất chia tách các điểm có y_i=1 và các điểm có y_i = -1. Mỗi siêu phẳng đều có thể được viết dưới dạng một tập hợp các điểm x thỏa mãn $w \cdot x - b = 0$,

Với mỗi i ta có:

$w \cdot x_i - b \geq 1$ cho x_i thuộc lớp thứ nhất hoặc

$w \cdot x_i - b \leq -1$ cho x_i thuộc lớp thứ hai.

2. Báo cáo kết quả

Các bước trong một quá trình xử lý dữ của model SVM:

- Tải dữ liệu: tải những dữ liệu có sẵn dùng để train hay test cho model.
- Tiền xử lý: tùy thuộc vào loại dữ liệu, bạn có thể cần xử lý trước dữ liệu trước khi đưa vào mô hình SVM.
- Tách dữ liệu: chia dữ liệu ra thành train và test.
- Chuyển đổi dữ liệu: chuyển đổi dữ liệu thành định dạng có thể được sử dụng làm đầu vào cho mô hình.
- Huấn luyện mô hình: dùng dữ liệu để huấn luyện mô hình.
- Đánh giá mô hình: đánh giá mô hình SVM dựa trên dữ liệu thử nghiệm.

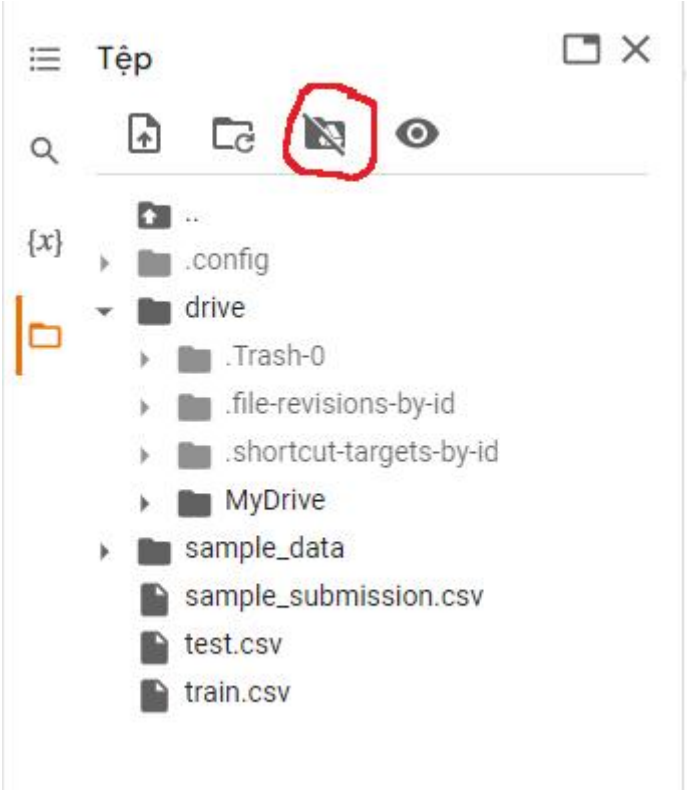
Kết quả:

Về phần tải dữ liệu, sử dụng hàm dưới để upload lên file csv

```
[3] from google.colab import files

    uploaded = files.upload()
```

Ngoài ra về phần dữ liệu ảnh, em để trên drive và sử dụng trực tiếp tool của google colab để add vào




```
[4] test_data = pd.read_csv('test.csv')
    train_data = pd.read_csv('train.csv')
```

Sau khi đã upload được dữ liệu thì em tiến hành xử lý các dữ liệu này.

```
[7] class_names=train_data.loc[:,'healthy':].columns
    print(class_names)

    number=0
    train_data['label']=0
    for i in class_names:
        train_data['label']=train_data['label'] + train_data[i] * number
        number=number+1
```

```
Index(['healthy', 'multiple_diseases', 'rust', 'scab'], dtype='object')
```

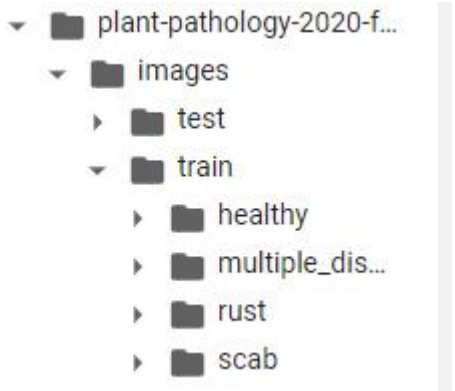
```
[18] train_data
```

	image_id	healthy	multiple_diseases	rust	scab	label
0	Train_0	0	0	0	1	3
1	Train_1	0	1	0	0	1
2	Train_2	1	0	0	0	0
3	Train_3	0	0	1	0	2
4	Train_4	1	0	0	0	0
...	...	...	...	...	...	...
1816	Train_1816	0	0	0	1	3
1817	Train_1817	1	0	0	0	0
1818	Train_1818	1	0	0	0	0
1819	Train_1819	0	0	1	0	2
1820	Train_1820	0	0	0	1	3

1821 rows × 6 columns

Em thêm 1 column ‘label’ để gán giá trị lưu giữ label của thông tin ảnh train. Nếu healthy là 1 thì label sẽ có giá trị là 0, nếu multiple_diseases là 1 thì label sẽ lưu giá trị là 1,tương tự tăng dần với các column sau.

Lý do em muốn lưu như vậy để tổ chức lại files ảnh tải lên thành như sau:



```
DIR= r'/content/drive/MyDrive/NHANDANG/plant-pathology-2020-fgvc7/images/train/'

def create_train_path():
    images=natsort.natsorted(os.listdir(DIR))
    for img in tqdm(images):
        label=get_label_img(img)
        path=os.path.join(DIR,img)

        if search("Train",img):
            if (img.split("_")[1].split(".")[0]) and label.item()==0:
                shutil.copy(path,r'/content/drive/MyDrive/NHANDANG/plant-pathology-2020-fgvc7/images/train/healthy')

            elif(img.split("_")[1].split(".")[0]) and label.item()==1:
                shutil.copy(path,r'/content/drive/MyDrive/NHANDANG/plant-pathology-2020-fgvc7/images/train/multiple_disease')

            elif(img.split("_")[1].split(".")[0]) and label.item()==2:
                shutil.copy(path,r'/content/drive/MyDrive/NHANDANG/plant-pathology-2020-fgvc7/images/train/rust')

            elif(img.split("_")[1].split(".")[0]) and label.item()==3:
                shutil.copy(path,r'/content/drive/MyDrive/NHANDANG/plant-pathology-2020-fgvc7/images/train/scab')



#create_train_path()
```

```
[ ] def create_test_path():
        images=natsort.natsorted(os.listdir(DIR))
        for img in tqdm(images):
            label=get_label_img(img)
            path=os.path.join(DIR,img)

            if search("Test",img):
              shutil.copy(path,r'/content/drive/MyDrive/NHANDANG/plant-pathology-2020-fgvc7/images/test')

    #create_test_path()
```

Bây giờ em đi vào xử lý những dữ liệu ảnh để có thể áp dụng vào model SVM.

Việc đầu tiên ta cần làm là load lên được những bức ảnh. Về xử lý ảnh này em sử dụng thư viện cv2 để hỗ trợ. Ví dụ ta có 1000 bức ảnh dưới dạng nxn pixel thì ảnh sẽ được tải lên trong shape(1000, n, n, 3) với 3 là miền thể hiện màu của ảnh dưới kênh màu RGB.

SVM được thiết kế để tìm ra đường biên (hay siêu phẳng) tốt nhất để phân tách các điểm dữ liệu của các lớp khác nhau trong không gian đa chiều. Vì vậy việc ta cần làm là điều chỉnh dữ liệu này thành dạng 1D làm dữ liệu đầu vào. Mỗi bức ảnh được làm phẳng thắng dạng 1D sẽ có shape: (1, n x n x 3).

Việc tiếp theo là chuẩn hóa các pixel thành (0,1) (từ hệ RGB sang gray) và lưu trữ dưới dạng array.

```
[12] from skimage.io import imread
     from skimage.transform import resize

     target = []
     images = []
     flat_data = []
     datadir = r'/content/drive/MyDrive/NHANDANG/plant-pathology-2020-fgvc7/images/train/'
     categories=['healthy','multiple_disease','rust','scab']

     for i in categories:
       class_num = categories.index(i)
       path=os.path.join(datadir,i)    #tạo path
       for img in os.listdir(path):
         img_arr = cv2.imread(os.path.join(path,img))
         img_resize = cv2.resize(img_arr,(150,150))    # resize ảnh
         flat_data.append(img_resize.flatten())     # flat ảnh
         images.append(img_resize)
         target.append(class_num)
     flat_data=np.array(flat_data)
     target = np.array(target)
     images = np.array(images)
```

Lúc này tập flat_data: là tập những trích xuất đặc trưng, một mảng thông tin của ảnh đã được làm phẳng để sử dụng cho việc huấn luyện mô hình.

Targert: là nhãn, tương ứng với giá trị của column label trên ứng với mỗi ảnh.

Images: là tập ảnh mới đã được resize lại thành (150x150).

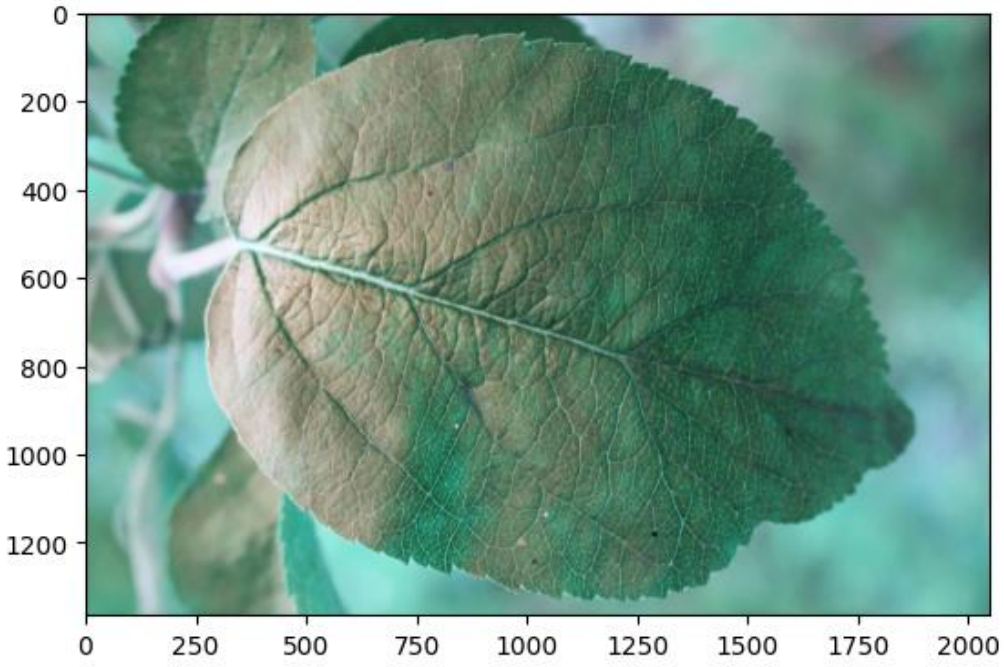


Image 1: ảnh ban đầu

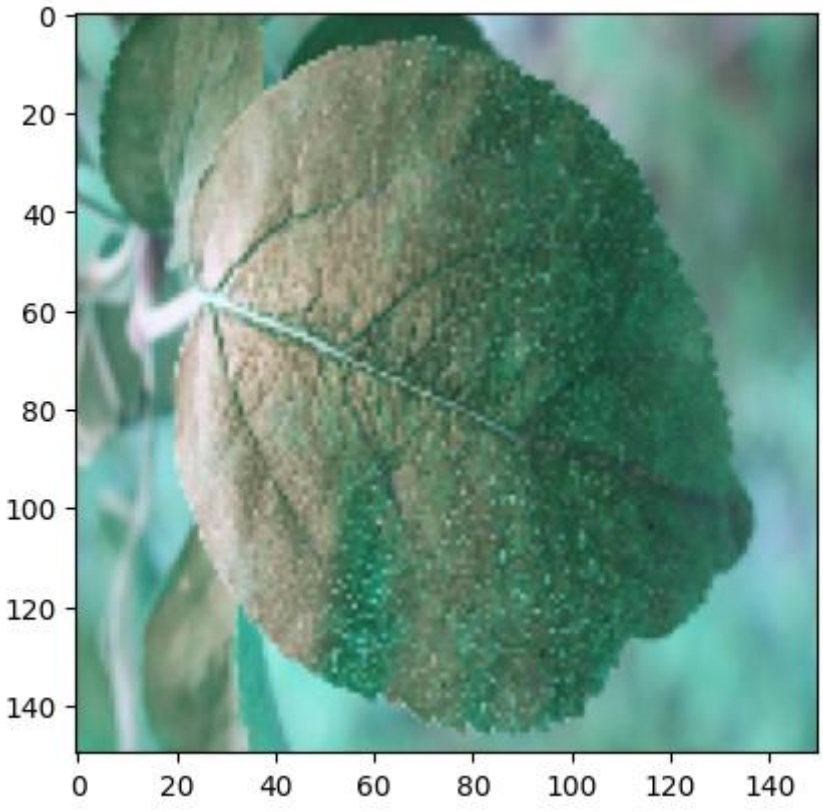


Image 2: ảnh sau khi resize

Việc resize này đươc thực hiện để có thể thu được những đặc trưng dễ dàng hơn. Ngoài ra các kích thước của ảnh đầu vào có thể sẽ khác nhau nên việc resize lại ảnh cũng là một bước quan trọng đối với tập dữ liệu đầu vào. Việc resize ảnh còn giúp cho mô hình SVM hoạt động tốt hơn vì nó có thể giảm thiểu số lượng tham số của model, do đó model sẽ nhanh hơn và có độ chính xác cao hơn. Việc giảm thiểu số lượng pixel của ảnh (kích thước file) cũng giúp giảm bộ nhớ cần thiết và làm cho việc xử lý dữ liệu trở nên dễ dàng hơn.

Bước tiếp theo ta sẽ tạo tập dữ liệu để huấn luyện model, trong bài em sử dụng hàm train_test_split của thư viện sklearn:

```
from sklearn.model_selection import train_test_split
X_train,x_test,y_train,y_test = train_test_split(flat_data,target,test_size=0.2,random_state=77,stratify=target)
```

Test_size=0.2, chia tập dữ liệu từ flat_data thành 80 train-20 test.

Bước vào phần huấn luyện model:

```
from sklearn.svm import SVC
svc = SVC(kernel='linear',gamma=0.001, probability=True)
svc.fit(X_train, y_train)
```

```
SVC
SVC(gamma=0.001, kernel='linear', probability=True)
```

Như đã nói ở trên em chọn SVM là model để xử lý thông tin ảnh trên. Model được lấy ra trong thư viên sklearn.svm.

Trong model trên em sử dụng kernel = ‘linear’, tức là sử dụng kernel tuyến tính để biến đổi các tập dữ liệu ban đầu sang không gian nhiều chiều hơn. Có nghĩa là tập dữ liệu ảnh ban đầu X_train sẽ được biến đổi ánh xạ tuyến tính sang miền mới có giá trị tương ứng. Hàm kernel tuyến tính thường được sử dụng trong SVM khi dữ liệu có thể được phân tách bằng một đường thẳng hoặc siêu phẳng.

Gamma được sử dụng để định biên độ ảnh hưởng giữa các điểm dữ liệu đến siêu phẳng. Trong đây em để giá trị gamma = 0.001 là khá nhỏ để các điểm được tính sự ảnh hưởng của các điểm dữ liệu là không lớn so với siêu phẳng. Các dữ liệu đầu vào X_train sẽ có tác động tương đối giống nhau với siêu phẳng.

Probability = True ở đây là cho phép áp dụng phép tính xác suất để model có thể dự đoán xác suất của mỗi điểm đầu vào. Còn nếu không để False thì sẽ không tính toán xác suất dự đoán mà chỉ đưa ra kết quả phân loại dựa trên một ngưỡng quyết định (decision threshold).

```
[ ]  y_pred = svc.predict(x_test)
```

```
[ ]  y_pred_prob= svc.predict_proba(x_test)
```

```
[ ]  from sklearn.metrics import accuracy_score
     print(f"The model is {accuracy_score(y_pred,y_test)*100}% accurate")
```

```
     The model is 38.139534883720934% accurate
```

```
[ ]  from sklearn.metrics import classification_report
     print(classification_report(y_test,y_pred))
```

```
                  precision    recall  f1-score   support

               0       0.40      0.37      0.39        62
               1       0.00      0.00      0.00        10
               2       0.35      0.32      0.33        73
               3       0.43      0.51      0.47        70

        accuracy                           0.38       215
       macro avg       0.30      0.30      0.30       215
    weighted avg       0.38      0.38      0.38       215
```

Sau khi split được dữ liệu và tiến hành huấn luyện model, và sử dụng model vào test thì hiệu quả đạt được của độ chính xác accuracy chỉ đạt cao nhất hơn 39% (em đã tham khảo các cách xử lý dữ liệu khác trên github).

Theo em nghĩ, các lý do dẫn đến việc độ chính xác thấp vì:

- Bước tiền xử lý ảnh chưa được áp dụng vào trong dữ liệu đầu vào (ảnh bị mờ, các bóng râm, nhiều, …)
- Tập dữ liệu đầu vào chưa đủ lớn để mô hình có thể học tốt.
- Cây có thể bị nhiều loại bệnh một lúc, tuy nhiên, đầu ra dự đoán chỉ là đúng là 1 loại bệnh nào (output chỉ là một giá trị target (là chỉ 1 giá trị từ 0 đến 4) đó dẫn đến việc chưa đạt được kết quả chính xác khi chỉ mới dự đoán được 1 loại bệnh chứ chưa dự đoán ra bệnh khác.
- Mô hình này chưa đủ thích hợp trên tập dữ liệu đầu vào.

Tiến vào bước cuối cùng là kiểm thử mô hình đã tạo với tập dữ liệu ảnh test đã được cho.

```
[30] target = []
     images = []
     flat_data = []
     title_img=[]
     datadir = r'/content/drive/MyDrive/NHANDANG/plant-pathology-2020-fgvc7/images/test/'



     path=os.path.join(datadir)    #tạo path
     for img in os.listdir(path):
       title_img.append(img.title()) #luu lai ten anh.jpg
       img_arr = cv2.imread(os.path.join(path,img))
       img_resize = cv2.resize(img_arr,(150,150))    # resize ảnh
       flat_data.append(img_resize.flatten())     # flat ảnh
       images.append(img_resize)
     title_img=np.array(title_img)
     flat_data=np.array(flat_data)
     images = np.array(images)
```

```
[31] title_img

     array(['Test_821.Jpg', 'Test_822.Jpg', 'Test_823.Jpg', ...,
            'Test_818.Jpg', 'Test_819.Jpg', 'Test_820.Jpg'], dtype='<U13')
```

Tiến hành xử lý ảnh như bước trên ta đã thực hiện và lưu lại tên của các ảnh.

```
[22] y_pred = svc.predict(flat_data)
```

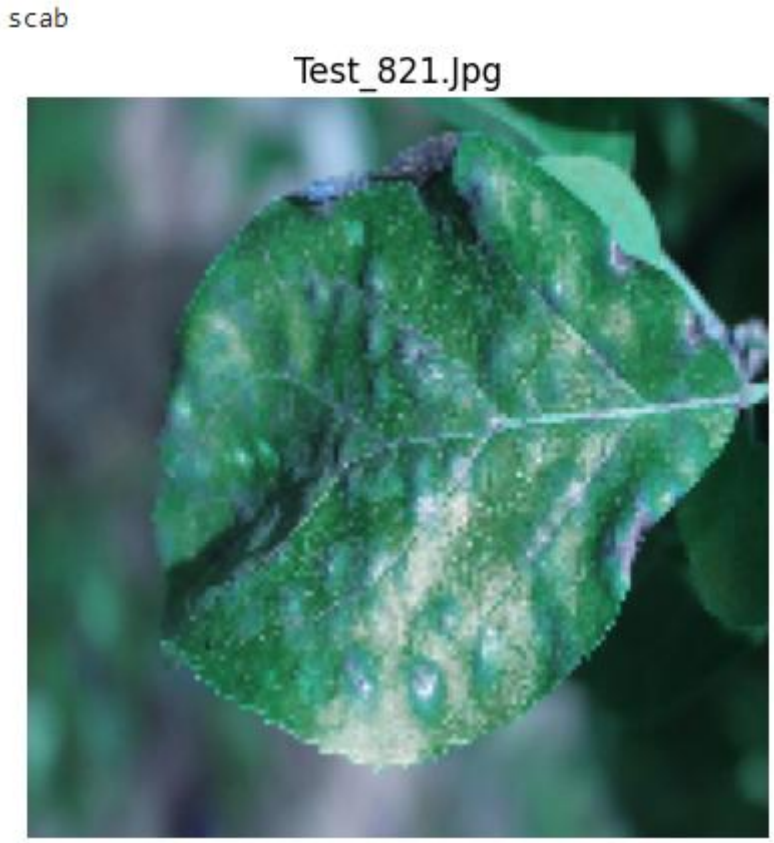
```
[23] y_pred

     array([3, 2, 3, ..., 0, 2, 3])
```

Tạo file csv cho kết quả trên

```
[32]
     yprednew=[]
     for i in y_pred:
         if i==0:
             yprednew.append("healthy")
         if i==1:
             yprednew.append("multiple_disease")
         if i==2:
             yprednew.append("rust")
         if i==3:
             yprednew.append("scab")
     ytestnew=np.array(yprednew)
     final=np.column_stack((title_img,ytestnew))
     df=pd.DataFrame(final,columns=['ImageId','predic'])
     df.head()
     df.to_csv('predict.csv',index=False)
```

Ảnh trên là kết quả cuối cùng, ta tạo một file csv để lưu lại những kết quả. Ảnh dưới là một ví dụ:



So sánh kết quả:

Sổ tay predict.csv ×

1 to 10 of 1821 entries Filter

ImageId	predic
Test_0.Jpg	scab
Test_1.Jpg	scab
Test_10.Jpg	scab
Test_100.Jpg	healthy
Test_1000.Jpg	healthy
Test_1001.Jpg	healthy
Test_1002.Jpg	rust
Test_1003.Jpg	rust
Test_1004.Jpg	healthy
Test_1005.Jpg	rust

Show 10 per page

1 2 10 100 180 183

Image 3: kết quả predict của mô hình

	A	B	C	D	E
1	image_id	healthy	multiple_d	rust	scab
2	Test_0	0.25	0.25	0.25	0.25
3	Test_1	0.25	0.25	0.25	0.25
4	Test_2	0.25	0.25	0.25	0.25
5	Test_3	0.25	0.25	0.25	0.25
6	Test_4	0.25	0.25	0.25	0.25
7	Test_5	0.25	0.25	0.25	0.25
8	Test_6	0.25	0.25	0.25	0.25
9	Test_7	0.25	0.25	0.25	0.25
10	Test_8	0.25	0.25	0.25	0.25
11	Test_9	0.25	0.25	0.25	0.25
12	Test_10	0.25	0.25	0.25	0.25

Image 4: kết quả file sample_submission

Kết luận

Qua so sánh giữa thực nghiện và bảng thì ta thấy được kết quả dự đoán trong bảng sample_submission.csv có thể cho đầu ra dự đoán được nhiều loại bệnh cùng lúc trên 1 ảnh còn thực nghiệm chỉ cho ra 1 kết quả duy nhất về 1 loại bệnh. Việc này làm rõ được một phần lý do vì sao mức độ accuracy chỉ đạt cao nhất hơn 39%.