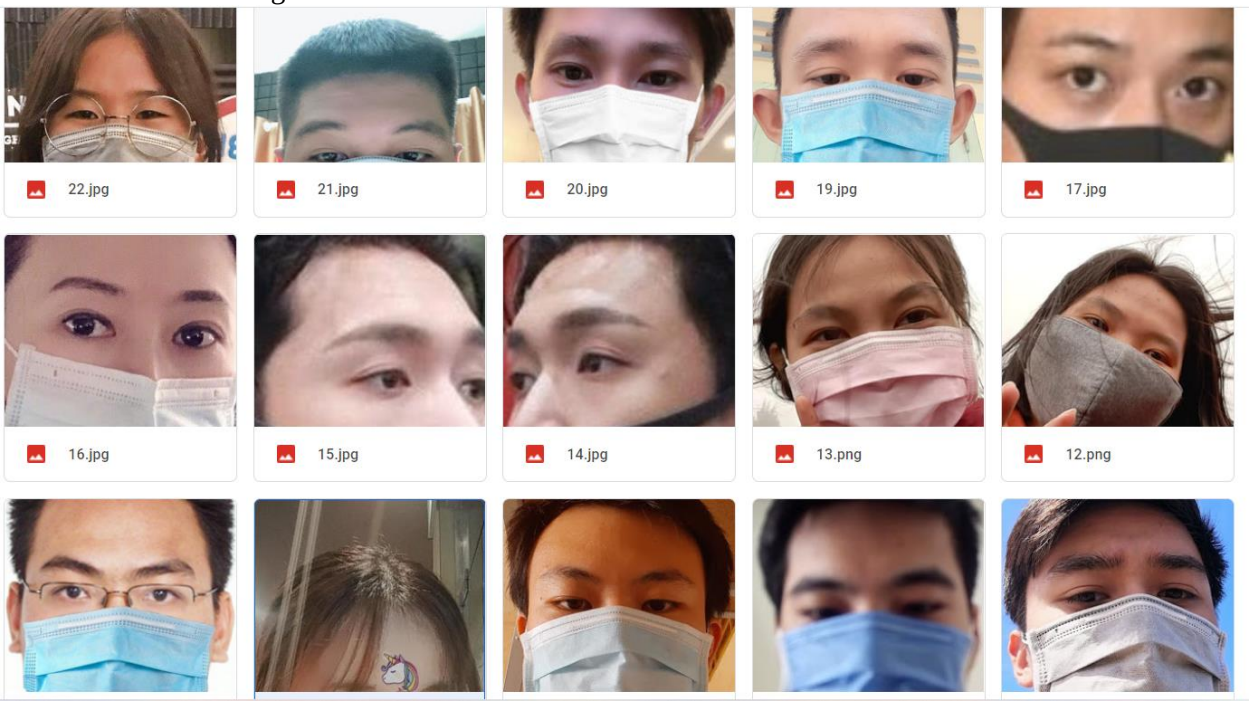
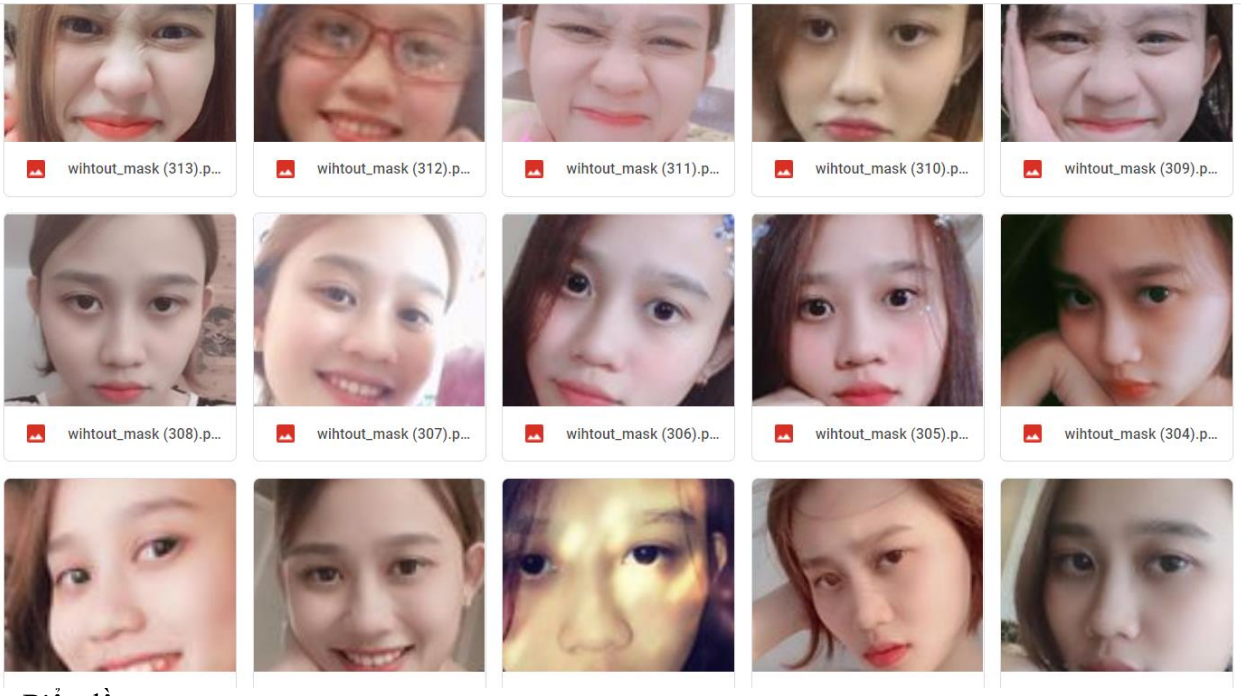
BÁO CÁO ĐỒ ÁN CUỐI KỲ

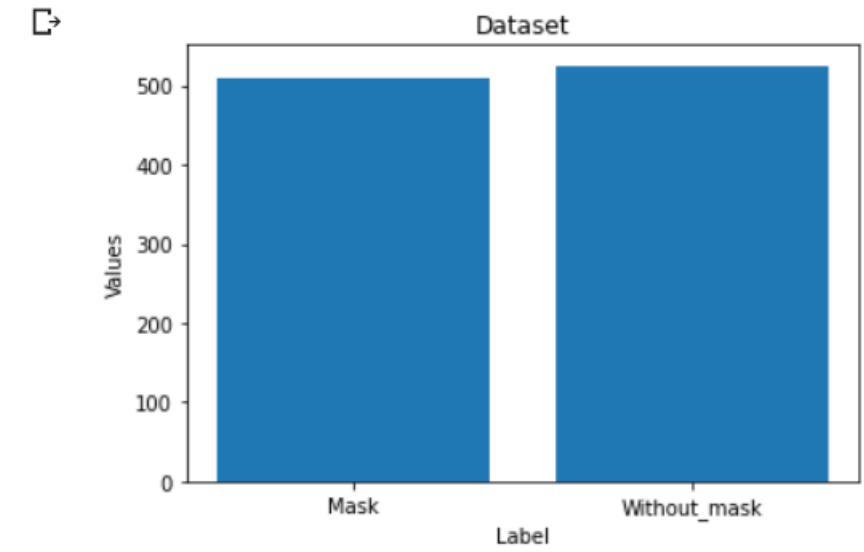
1. **Mô tả bài toán**
   a. Bài toán : Nhận diện ảnh khuôn mặt một người đeo khẩu trang
   b. Lí do chọn bài toán :
   + COVID-19 , bệnh viêm đường hô hấp cấp do chủng mới của virus corona được phát hiện lần đầu tiên tại thành phố Vũ Hán , tỉnh Hồ Bắc , Trung Quốc vào tháng 12 / 2019.
   + Đến nay, có 215 quốc gia / vùng lãnh thổ trên toàn câu ghi nhân trường hợp mắc . Trong đó nước Mỹ là nước có người nhiễm nhiều nhất.
   + Hiện tại sau 99 ngày Việt Nam không có ca nhiễm trong cộng đồng thì dịch đang dần trở lại.
   + Cách để phòng dịch tốt nhất được chính phủ và WHO khuyến cáo là đeo khẩu trang khi ra đường hay tiếp xúc với người khác.
   ⇨ Chính vì thế nhóm đã lên ý tưởng xây dựng một mô hình tự động nhận diện xem mọi người có đeo khẩu trang hay không.
   ⇨ Bài toán này là bài toán cơ sở , khi kết hợp với bài toán nhận diện khuôn mặt người sẽ tạo thành một bài toán thực tế có thể áp dụng trên diện rộng.
   c. Mô tả:
   Input : hình ảnh khuôn mặt một người
   Ouput :
   + 1 : người này có đeo khẩu trang
   + 0 : người này không đeo khẩu trang
2. **Mô tả về bộ dữ liệu**
   - Các ảnh trong bộ data được thu thập từ các hình ảnh tự chụp , xin ảnh người quen từ các ảnh chụp trước và phần lớn data là ảnh được thu thập từ Google.
   - Ảnh sau khi được thu thập sẽ dùng tool CasadeClassifer của OpenCV cắt mặt ra hoặc tự cắt đối với các mặt tool không nhận được mặt
   - Sau khi cắt mặt từ bộ ảnh thu được :
   + 509 ảnh có khẩu trang



+ 525 ảnh không có khẩu trang



+ Biểu đồ :



⇨ Bộ data khá cân bằng và khá đa dạng với các gốc mặt khác nhau, các loại khẩu trang khác nhau. Tuy nhiên số lượng ảnh trong dataset còn khá ít.

- Tiền xử lí dữ liệu:
+ Vì ảnh còn khá ít nên nhóm quyết đinh sử dụng hàm ImageDataGenerator của thư viện Keras để tăng dữ liệu cho bộ dataset

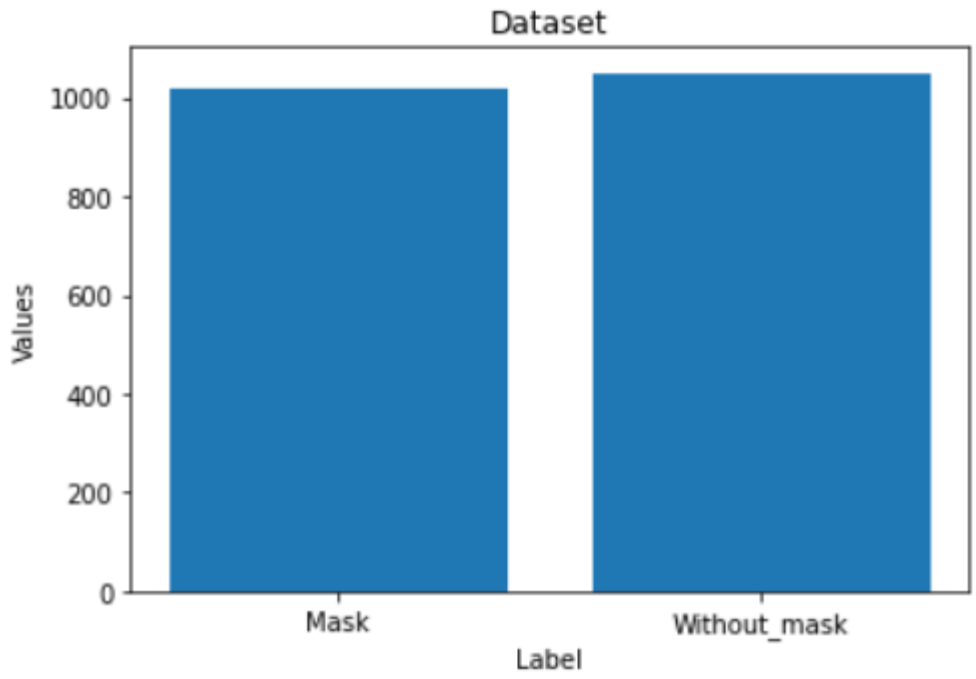
Fine-turning hàm ImageDataGenerator

- rotation_range : góc quay (0 - 20)
- zoom_range : độ phóng (0 - 0.15)
- width_shift_range, height_shift_range : độ dịch ảnh (0 - 0.2)
- horizontal_flip : lật ảnh theo chiều ngang

In [203]:
```
# Sử dụng kỹ thuật Data Augmentation để tăng kích thước cho bộ train data
aug = ImageDataGenerator(
      rotation_range=20,
      zoom_range=0.15,
      width_shift_range=0.2,
      height_shift_range=0.2,
      horizontal_flip=True )
```

+ Sau khi sử dụng hàm ImageDataGenerator từ 1 ảnh được load ban đầu sẽ trở thành 2 ảnh khác nhau

+ Biểu đồ dữ liệu sau khi sử dụng ImageDataGenerator



+ Ảnh sau khi được load lên sẽ được resize về kích thước 32x32 nhằm giảm số lượng feature cho model

+ Sau đó ảnh từ dạng mảng 3 chiều sẽ được làm phẳng thành 1 Vector

```
# load hình ảnh
image = cv2.imread(imagePath)

# preprocess ảnh
image = cv2.resize(image,(32,32)).flatten()
```

+ Cuối cùng các Vector ảnh là label của nó được chuyển sang dạng numpy

```
# chuyển label và data sang dạng mảng
labels = np.array(labels)
data = np.array(data)
```

- Tiến hành chia dữ liệu thành 2 phần : 80% dùng để train, validation và 20% dùng để test

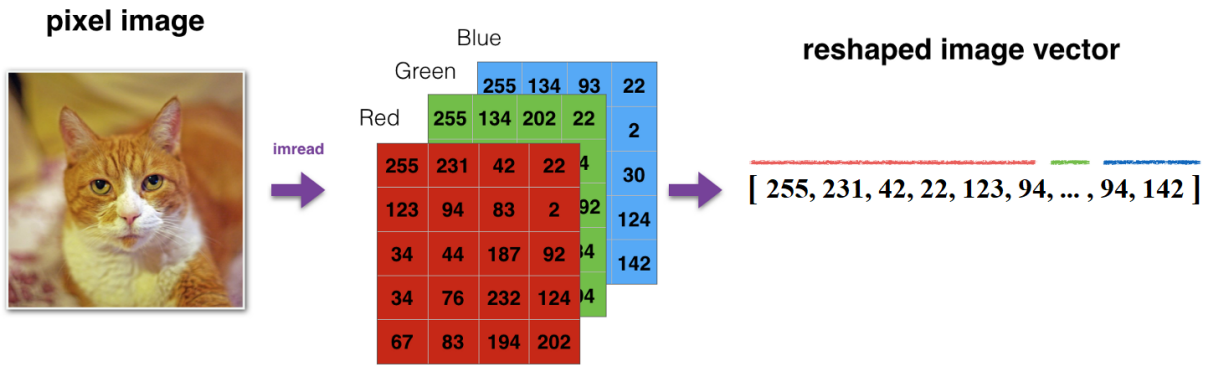
```
# chia dữ liệu thành 80% để train, validation bằng K-Fold và 20% để test
(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.2,stratify=labels,random_state=1)
```

3. Mô tả về đặc trưng

- Biến đổi từ 1 bức ảnh có mảng 3 chiều thành 1 vector
- Từ một ảnh có kí thước 32x32 thì sẽ chuyển thành 1 vector có 32x32x3 phần tử vì một ảnh có 3 kênh màu Red , Green , Blue.
- Việc này được thực hiện thông qua hàm flatten của thư viện numpy

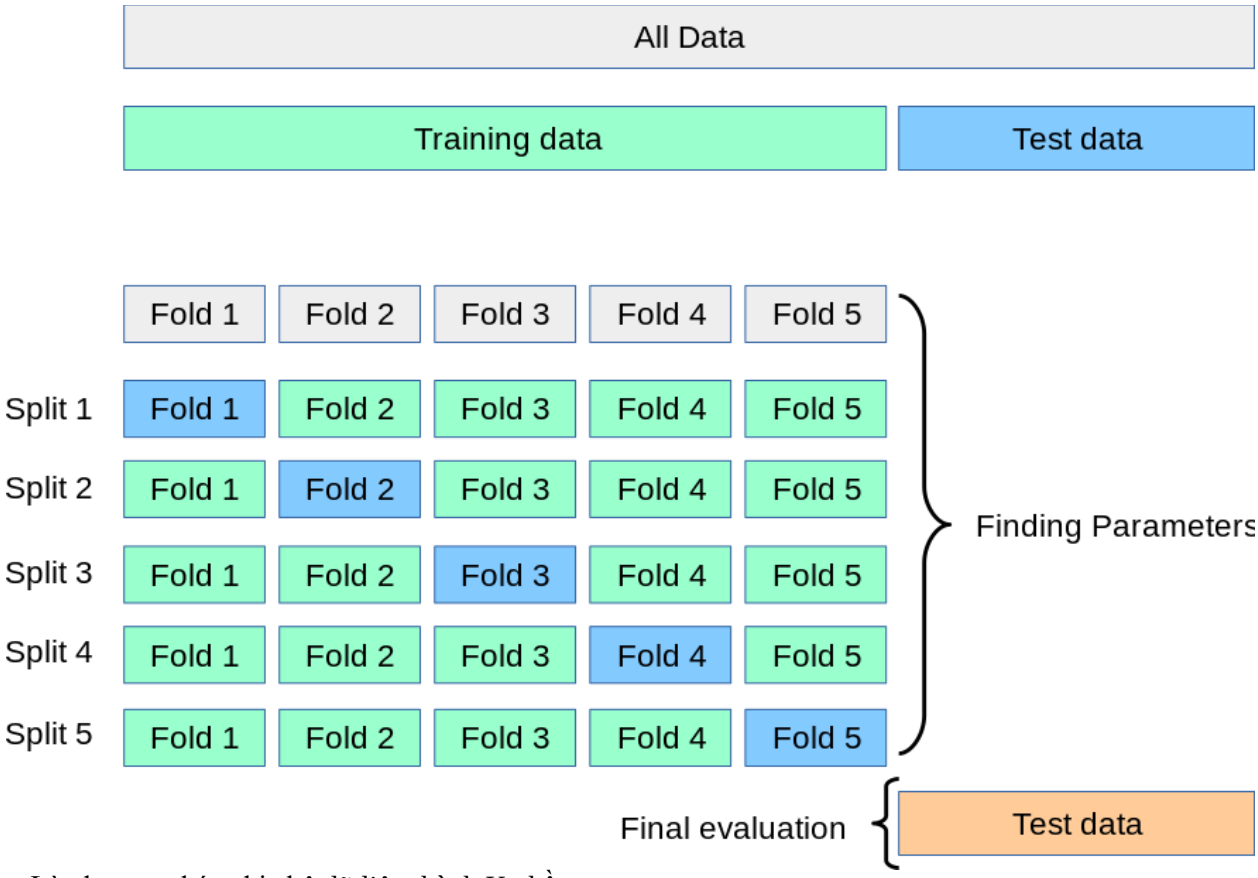
```
# preprocess ảnh
image = cv2.resize(image,(32,32)).flatten()
```

- Hàm flatten sẽ lấy hàng ngang của từng cột sau đó ghép nó thành 1 hàng ngang duy nhất



4. Mô tả về thuật toán

- Do label có dạng binary : 0 và 1 vì thế chọn các thuật toán Binary Classification
- Các model được sử dụng:
  + Decision Tree
  + Random Forest
  + Logictic Regression
- Do dataset khá ít (khoảng 2000 ảnh sau khi được tăng) , vì thế sử dụng phương pháp Cross-Validation để tiến hành đánh giá các model được chọn . Cụ thể trong bài toán này sẽ sử dụng phương pháp k-fold.
- Phương pháp K-Fold :



+ Là phương pháp chia bộ dữ liệu thành K phần
+ Sẽ có K lần training
+ Trong mỗi lần train, 1 phần sẽ được dùng để đánh giá và K – 1 phần còn lại dùng để training cho model
+ Độ chính xác của model sau khi sử dụng k-fold là trung bình cộng của mỗi lần train

⇨ Cụ thể trong bài toán này ta sẽ sử dụng 10 – Fold (chia thành 10 phần)

- Training và Validation sử dụng k-fold sau đó chọn ra model có accuracy cao nhất
- Tiến hành training và đánh giá model được chọn thông qua hàm confusion_matrix và classification_report của thư viện sklearn

5. Đánh giá kết quả, kết luận

- Sử dụng K – Fold để đánh giá 3 model được chọn :

```
# Xuất ra giá trị trung bình của results
print('%s: %f' % (name, cv_results.mean()))
```

```
LR: 0.850628
DT: 0.813782
RF: 0.906261
```

⇨ Có thể thấy sau khi sử dụng k-fold để training và đánh giá thì model Random Forest có độ chính xác cao nhất. Vì thế , ta sẽ chọn model này để đánh giá trên tập test .

- Tiến hành đánh giá trên tập test với Random Forest:

```
# Đánh giá model
print(confusion_matrix(testY, predictions))
print(classification_report(testY, predictions))
```

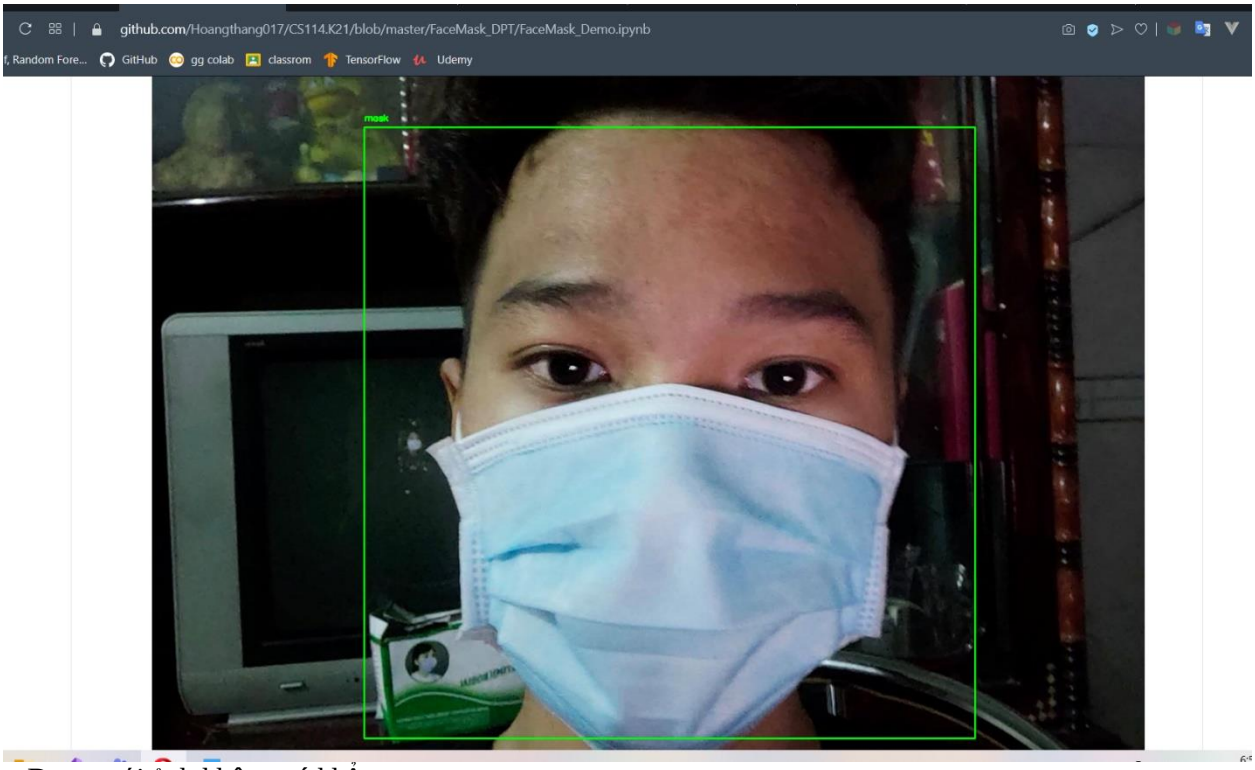
```
[[200  10]
 [ 30 174]]
              precision    recall  f1-score   support

           0       0.87      0.95      0.91       210
           1       0.95      0.85      0.90       204

    accuracy                           0.90       414
   macro avg       0.91      0.90      0.90       414
weighted avg       0.91      0.90      0.90       414
```

- Đánh giá :
  + Độ chính xác của Random Forest khá cao : 90%
  + Độ chính xác của tập train và tập test gần như bằng nhau nên model có thể sử dụng được với lỗi không quá lớn
  + Như có thể thấy ở Confusion_matrix :
    - Model đều dự đoán khá chính xác ở trên cả 2 label
    - Giá trị của False Positive cao hơn False Negative , tuy nhiên ở bài toán này ta có thể chấp nhận được lỗi này vì thiệt hại của nó không quá lớn. Vì khi dự đoán sai 1 người đeo khẩu trang thành không đeo khẩu trang sẽ ít thiệt hại hơn là dự đoán 1 người không đeo khẩu trang thành có đeo khẩu trang.

- Sử dụng model để dự đoán với ảnh thực tế :
  + Demo với ảnh có khẩu trang :

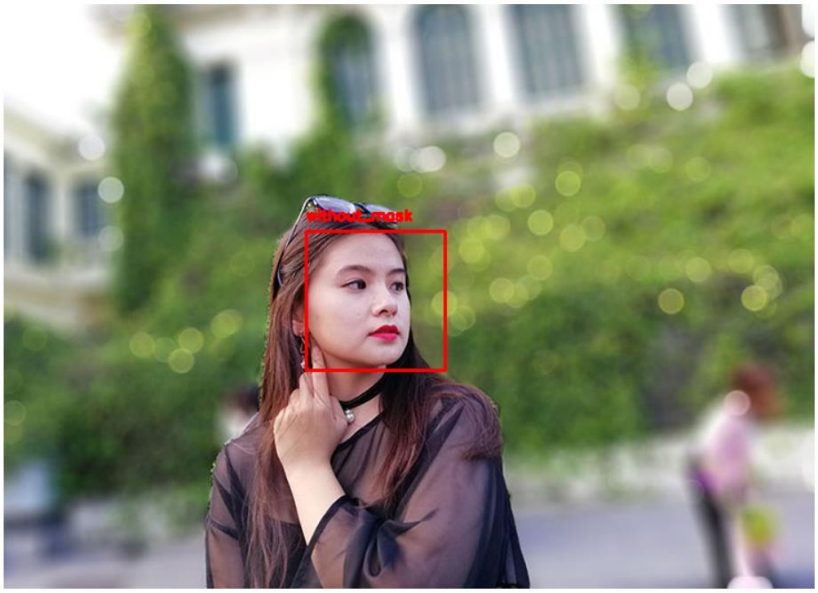


+ Demo với ảnh không có khẩu trang :

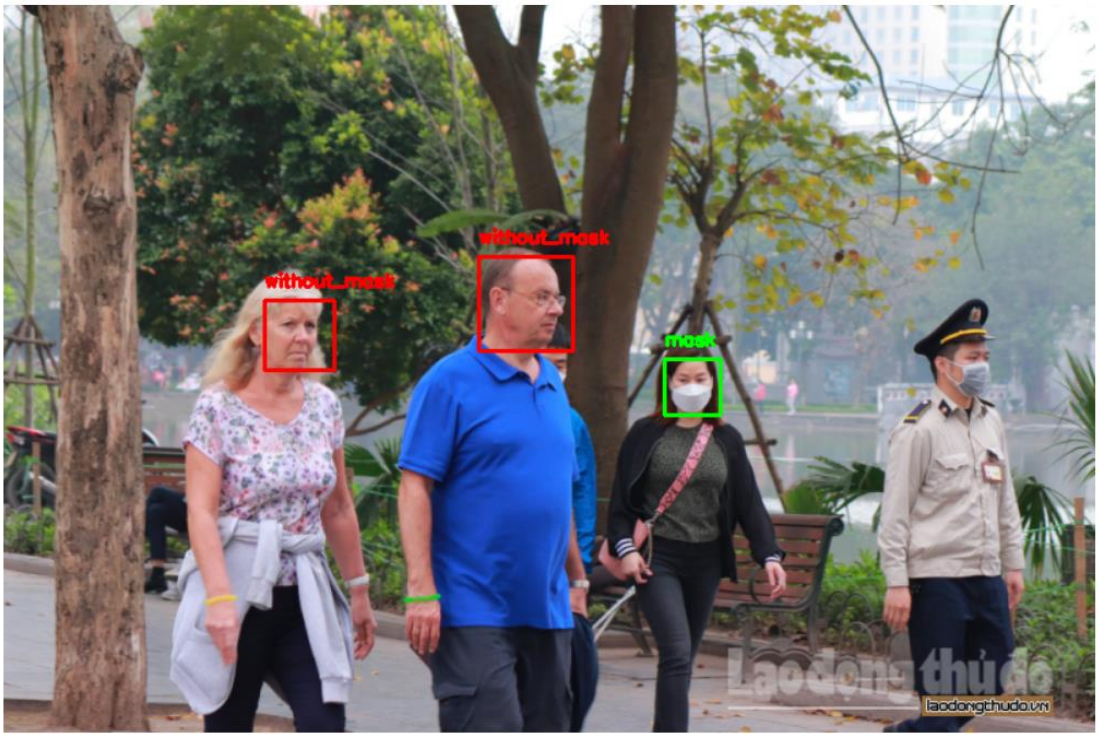
Demo chân dung một người không đeo khẩu trang

In [ ]:
```
# Load hình
img = cv2.imread("/content/img2.jpg")

# Dự đoán
Predict_FaceMask(img)
```



+ Demo với ảnh nhiều người :



+ Tiến hành cắt mặt trong ảnh trên mà tool không nhận diện được để tiến hành test với model

```
# Load hình
face = cv2.imread("/content/cut _ img tool no detected.png")

cv2_imshow(cv2.resize(face,(224,224)))

face = cv2.resize(face,(32,32)).flatten()
face = np.array(face).reshape(1,-1)

result = Model_LR.predict(face)

if (result == 0 ):
  print("------------------------------Wihtout_Mask------------------------------")
else:
  print("------------------------------Mask------------------------------")
```

----------Mask----------

- Kết luận :
  + Model cũng dự đoán chính xác trên cả ảnh thực tế
  + Tuy nhiên , để đánh giá được một bức ảnh nhiều người model phải phụ thuộc vào tool nhận diện khuôn mặt của openCV
  + Model vẫn còn một số hạn chế như việc nhầm lẫn các vật thể tương tự che vào mặt như khẩu trang model sẽ cho kết quả không chính xác

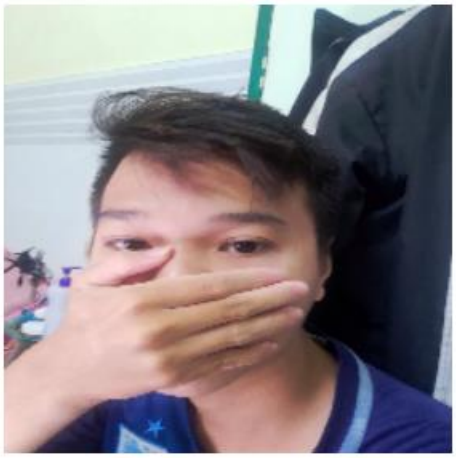
```
# Load hình
face = cv2.imread("/content/example (19).jpg")

cv2_imshow(cv2.resize(face,(224,224)))

face = cv2.resize(face,(32,32)).flatten()
face = np.array(face).reshape(1,-1)

result = Model_LR.predict(face)

if (result == 0 ):
  print("------------------------------Wihtout_Mask------------------------------")
else:
  print("------------------------------Mask------------------------------")
```



------------------------------Mask------------------------------

6. Link tham khảo

https://towardsdatascience.com/classifying-cat-pics-with-a-logistic-regression-model-e35dfb9159bb
https://numpy.org/doc/stable/reference/generated/numpy.ndarray.flatten.html
https://medium.com/@alkeshab/face-detection-using-opencv-in-google-colaboratory-a7529a2bb921
https://scikit-learn.org/stable/modules/cross_validation.html