TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

KHOA KHOA HỌC MÁY TÍNH

MACHINE LEARNING

Đề Tài: **NHẬN DIỆN GIỚI TÍNH**

GV: PGS.TS Lê Đình Duy –
THS. Phạm Nguyễn Trường An

Lớp: CS114.K21

Nhóm: Trần Quốc Toàn 18520384

Nguyễn Tri Đức Anh 18520384

Trần Văn Thục 18520384

MỤC LỤC

CHƯƠNG I: GIỚI THIỆU ĐỒ ÁN

1. TẦM QUAN TRỌNG CỦA NHẬN DIỆN CÁC ĐẶC TÍNH TRÊN KHUÔN MẶT

Ngày nay, việc khuôn mặt được sử dụng rất rộng rãi trong nhiều lĩnh vực khác nhau. Dựa vào dữ liệu khuôn mặt thu thập được áp dụng rộng rãi trong vô số ứng dụng từ bảo mật đến quảng cáo:

- Xây dựng các hệ thống nhận diện tội phạm, theo dõi, tìm kiếm … hay các nhà sản xuất điện thoại, các công ty doanh nghiệp dùng khuôn mặt để tăng tính bảo mật
- Tạo các ứng dụng(Face app- cho phép người dùng thay đổi đặc tính trên bức ảnh có khuôn mặt), MXH dùng để gán nhãn tên trong ảnh
- Sử dụng các đặc tính khuôn mặt để đưa ra quảng cáo, tiếp thị phù hợp với người dùng

2. MỤC ĐÍCH CHỌN ĐỀ TÀI

Giới tính được xem là một trong những đặc tính tiêu biểu trên khuôn mặt con người. Phát hiện khuôn mặt và nhận diện được giới tính của một người là bước nhỏ đầu tiên trước khi có thể nhận diện hết tất cả đặc tính của khuôn mặt để áp dụng vào cuộc sống

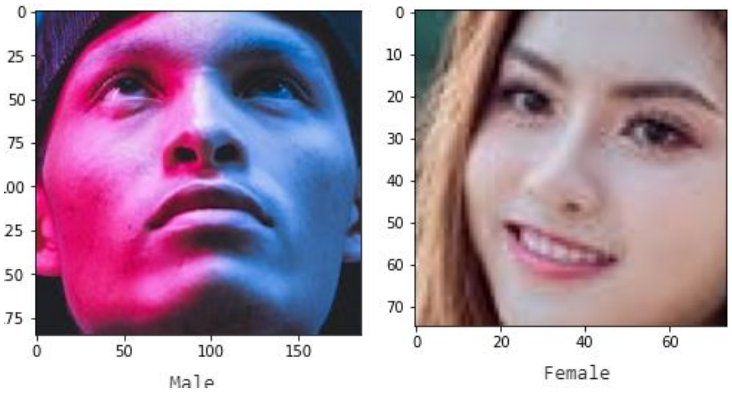
3. MÔ TẢ BÀI TOÁN

Chương trình sau khi cài đặt cho phép người sử dụng nhập vào 1 ảnh bất kì. Chương trình sẽ xác định được những khuôn mặt trong bức ảnh, đưa ra kết quả nhận diện giới tính cho từng khuôn mặt đó

Input: Bức ảnh chứa mặt của 1 hay nhiều người(nếu nhiều người thì không quá nhiều)

Output: khuôn mặt được nhận diện và kết quả nhận diện giới tính của từng khuôn mặt đó

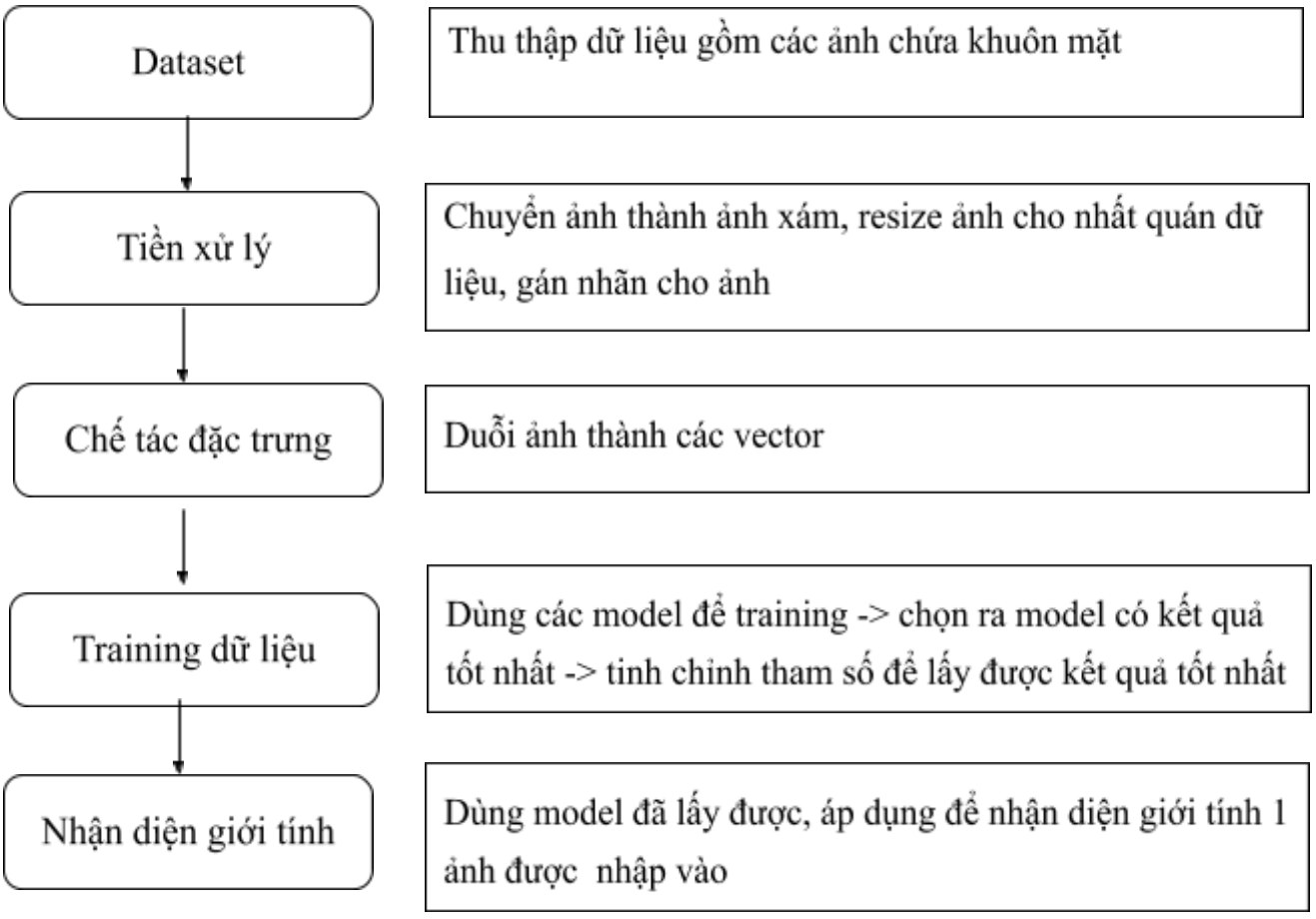
Input



Output

4. XÂY DỰNG CHƯƠNG TRÌNH NHẬN DIỆN GIỚI TÍNH

Các bước xây dựng chương trình nhận diện giới tính được tóm tắt như sau



CHƯƠNG 2: XÂY DỰNG DỮ LIỆU VÀ TIỀN XỬ LÝ

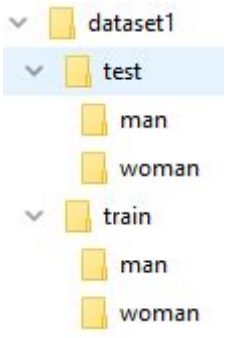
1. XÂY DỰNG DATASET KHUÔN MẶT NGƯỜI

Yêu cầu data cần thu thập: ảnh khuôn mặt nam hoặc nữ; độ phân giải vừa phải, dễ nhận biết giới tính

Thu thập data: Dataset được lấy từ 2 nguồn

Kaggle: xấp xỉ 2200 ảnh có cả nam lẫn nữ, đã được phân nhãn (có ghi chú trong link tham khảo)

Được chia sẻ trên https://viblo.asia : xấp xỉ 23000 có cả nam lẫn nữ (có ghi chú trong link tham khảo)



Cây thư mục lưu dataset từ kaggle

2. XỬ LÝ DỮ LIỆU

Từ 2 bộ dữ liệu ảnh khuôn mặt người trên, chọn lọc và tạo dataset có thể sử dụng:

- Chọn lọc ảnh: chỉ chọn ảnh có độ phân giải tương đối trở lên để đảm bảo độ chính xác cao(các công đoạn loại bỏ ảnh nhiễu, ảnh bị mờ, không rõ mặt… được bỏ qua vì dataset được chia sẻ đã làm qua bước này, chỉ chọn lọc ảnh độ phân giải tương đối trở lên để phù hợp bài toán) -> lược bỏ bớt dataset ở tập kaggle; thêm dataset từ Viblo vào các thư mục có sẵn trên(lấy ảnh nam và nữ với số lượng tương đối, tránh quá chênh lệch dữ liệu)

- Giữ nguyên tập train và test để sử dụng trong các bài toán khác
- Kết quả sau khi chọn lọc và phân loại: 16604 bức ảnh, trong đó 8336 bức ảnh nam và 8268 bức ảnh nữ

Các bước tiền xử lí ảnh:

- Chuyển tất cả ảnh thành màu xám
- Resize ảnh: bởi vì mỗi ảnh trong bộ dữ liệu có chất lượng và độ phân giải khác nhau nên cần đưa nó về cùng 1 size để nhất quán, thuận tiện trong việc xử lý
- Gán nhãn cho từng ảnh: 0 là nam và 1 là nữ. Gom vector
- Chế tác đặc trưng: duỗi ảnh thành từng vector

Chia lại bộ dữ thành tập train và test (với tỉ lệ tùy chọn, trong phần cài đặt là 20%) để thuận tiện cho việc training dữ liệu

CHƯƠNG 3: PHƯƠNG PHÁP XÂY DỰNG CHƯƠNG TRÌNH

1. SO SÁNH ĐỘ HIỆU QUẢ CỦA NHIỀU MODEL

Để tìm ra được 1 model phù hợp với yêu cầu bài toán đề ra và với bộ dữ liệu, dùng 1 số model sau để kiểm tra độ chính xác khi được training với bộ dữ liệu trên.

Một số model phù hợp được sử dụng là:

- Logistic Regression
- Linear Discriminant Analysis
- K Neighbors Classifier
- Decision Tree Classifier
- Gaussian NB
- SVC
- Perceptron

So sánh kết quả dựa vào độ chính xác(accuracy) của từng model và chọn model tốt nhất để đưa vào tinh chỉnh.

Tinh chỉnh tham số: tinh chỉnh các tham số ảnh hưởng đến độ chính xác của model -> so sánh kết quả trước và sau khi tinh chỉnh, chọn ra model phù hợp với kết quả tốt nhất để sử dụng nhận diện ảnh được nhập vào.

2. DÙNG FACE RECOGNITION ĐỂ TÌM KHUÔN MẶT TRONG ẢNH

Sau khi nhập ảnh cần nhận diện vào được xử lí như data để phù hợp cho việc test.

FACE RECOGNITION: được ứng dụng để: tìm khuôn mặt trong 1 bức hình, tìm và đánh dấu(hoặc thao tác) các đặc điểm trên khuôn mặt, nhận dạng khuôn mặt trong ảnh, hoặc có thể kết hợp với các thư viện khác của python để nhận dạng khuôn mặt trong thời gian thực hoặc video (nguồn dẫn trong link tham khảo)

Trong bài toán này chúng ta chỉ dùng ứng dụng tìm khuôn mặt trong 1 bức hình.

Lấy các khuôn mặt đã được xác định test với model đã được lưu và đưa ra kết quả dự đoán giới tính của khuôn mặt đó.

CHƯƠNG 4: CÀI ĐẶT CHƯƠNG TRÌNH

1. CÔNG CỤ THỰC HIỆN

Ngôn ngữ xử dụng: Python

Công cụ: Google Colab(được lưu trữ trên github)

2. CÀI ĐẶT CHƯƠNG TRÌNH

Import các thư viện

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as implt
import os
import seaborn as sns
import cv2 as cv
from PIL import Image
import warnings
```

Liên kết với google drive và giải nén data(data được lưu dạng zip trong drive)

```
from google.colab import drive
drive.mount('/content/drive')
!unzip '/content/drive/My Drive/DataSet_Male_female/Dataset_Male_female.zip'
```

Tạo tập train_path bao gồm tất cả các hình ảnh trong tập train

- train_man: tất cả ảnh male trong tập train
- train_woman: tất cả ảnh female trong tập train
- test_man: tất cả ảnh male trong tập test
- test_woman: tất cả ảnh female trong tập test

```
train_path = "/content/Dataset_Male_female/genderdetectionface/dataset1/train"
test_path = "/content/Dataset_Male_female/genderdetectionface/dataset1/test"

train_woman = sorted(os.listdir(train_path +'/woman'))
train_man =  sorted(os.listdir(train_path +'/man'))

test_woman = sorted(os.listdir(test_path +'/woman'))
test_man =  sorted(os.listdir(test_path +'/man'))
```

Các bước xử lí: chuyển ảnh thành ảnh xám, resize ảnh để nhất quán dữ liệu, gán nhãn

- Chuyển ảnh thành ảnh xám

```
faces = Image.open(train_path +'/woman/'+ i).convert('L')
```

- Resize ảnh

```
faces = faces.resize((img_size,img_size), Image.ANTIALIAS)
```

- Gán nhãn

```
label.append(1) #1 là nữ
label.append(0) #0 là nam
```

Các bước được xử lí trên tập train(tương tự với tập test)

```
img_size = 50
women_faces = []
men_faces = []
label = []

for i in train_woman:

        if os.path.isfile(train_path +'/woman/'+ i):
            faces = Image.open(train_path +'/woman/'+ i).convert('L') #Chuyển thành ảnh xám
            faces = faces.resize((img_size,img_size), Image.ANTIALIAS) #resizing thành 50,50
            faces = np.asarray(faces)/255.0
            women_faces.append(faces)
            label.append(1) #1 là nữ

for i in train_man:
        if os.path.isfile(train_path+'/man/'+ i):
            faces = Image.open(train_path+'/man/'+ i).convert('L')
            faces = faces.resize((img_size,img_size), Image.ANTIALIAS)
            faces = np.asarray(faces)/255.0
            men_faces.append(faces)
            label.append(0) #0 là nam
```

Gom các vector ảnh và nhãn thành 1 mảng(xử lý trên tập train, tương tự với tập test)

```
x_train = np.concatenate((women_faces,men_faces),axis=0) # Gom tất cả các vecto ảnh lại
x_train_label = np.asarray(label)# biến tập nhãn thành mảng label gồm 0 và 1
x_train_label = x_train_label.reshape(x_train_label.shape[0],1)
```

Gom nhóm dữ liệu nhãn và giá trị của nhãn

```
x = np.concatenate((x_train,x_test),axis=0)
y = np.concatenate((x_train_label,x_test_label),axis=0)
```

Chế tác đặc trưng: duỗi từng ảnh thành 1 vecto có kích thước 1x2500

```
x = x.reshape(x.shape[0],x.shape[1]*x.shape[2])
```

Phân chia lại tập train và test ngẫu nhiên(trong bài lấy 20% test từ tổng dataset)

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

Áp dụng các model để tính độ chính xác

- Import các model

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import Perceptron
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

Kết quả từng model:

- Logistic Regression

```
LR = LogisticRegression()
LR.fit(X_train,Y_train)
model_list.append(LR.score(X_test, Y_test))
predict=LR.predict(X_test)
print('Accuracy:  ',accuracy_score(Y_test,predict))
print('Classification report: \n',classification_report(Y_test,predict))
print('Confusion matrix: \n',confusion_matrix(Y_test,predict))
```

```
Accuracy:   0.7946401686239085
Classification report:
              precision    recall  f1-score   support

           0       0.79      0.80      0.80      1681
           1       0.80      0.79      0.79      1640

    accuracy                           0.79      3321
   macro avg       0.79      0.79      0.79      3321
weighted avg       0.79      0.79      0.79      3321

Confusion matrix:
 [[1349  332]
 [ 350 1290]]
```

Giải thích các tham số (lấy kết quả từ Logistic Regression và tương tự với các model khác)

- Accuracy: 0.7946 là độ chính xác và được tính là số lượng dự đoán đúng / tổng số dự đoán
- Classification report: trong tập test có 1681 ảnh nhãn 0 và 1640 ảnh nhãn 1. Trong đó f1-score của nhãn 0 và 1 tương ứng là 80% và 79% với

*f1-score=2 x (recall x precision) / (recall + precision) trong đó:

+ recall nhãn 0 = (số lượng ảnh nhãn 0 được model dự đoán chính xác) / (số lượng ảnh nhãn 0 trong thực tế)

+ precision nhãn 0 = (số lượng ảnh nhãn 0 được model dự đoán chính xác) / (số lượng ảnh nhãn 0 được model dự đoán)

- f1-score trung bình theo cách tính weighted avg (có nhân với tỷ lệ của số lượng nhãn) là 0.79. (0.79 = 0.8. (1681/3321) + 0.79. (1640/3321))

- Confsusion matrix:

+ model dự đoán đúng 1349 ảnh vốn có nhãn là 0

+ model dự đoán sai 350 ảnh vốn có nhãn là 1

+ model dự đoán sai 332 ảnh vốn có nhãn là 0

+ model dự đoán đúng 1290 ảnh vốn có nhãn là 1

- Linear Discriminant Analysis

```
LDA = LinearDiscriminantAnalysis()
LDA.fit(X_train,Y_train)
model_list.append(LDA.score(X_test, Y_test))
predict=LDA.predict(X_test)
print('Accuracy:  ',accuracy_score(Y_test,predict))
print('Classification report: \n',classification_report(Y_test,predict))
print('Confusion matrix: \n',confusion_matrix(Y_test,predict))
```

```
Accuracy:   0.7717554953327311
Classification report:
              precision    recall  f1-score   support

           0       0.76      0.79      0.78      1681
           1       0.78      0.75      0.76      1640

    accuracy                           0.77      3321
   macro avg       0.77      0.77      0.77      3321
weighted avg       0.77      0.77      0.77      3321

Confusion matrix:
 [[1333  348]
 [ 410 1230]]
```

- K Neighbors Classifier

```
from sklearn.neighbors import KNeighborsClassifier
k= KNeighborsClassifier()
k.fit(X_train,Y_train)
k.score(X_test, Y_test)
model_list.append(k.score(X_test, Y_test))
predict=k.predict(X_test)
print('Accuracy:  ',accuracy_score(Y_test,predict))
print('Classification report: \n',classification_report(Y_test,predict))
print('Confusion matrix: \n',confusion_matrix(Y_test,predict))
```

```
Accuracy:   0.7886178861788617
Classification report:
              precision    recall  f1-score   support

           0       0.77      0.82      0.80      1681
           1       0.81      0.75      0.78      1640

    accuracy                           0.79      3321
   macro avg       0.79      0.79      0.79      3321
weighted avg       0.79      0.79      0.79      3321

Confusion matrix:
 [[1383  298]
 [ 404 1236]]
```

- Decision Tree Classifier

```
D = DecisionTreeClassifier()
D.fit(X_train,Y_train)
D.score(X_test, Y_test)
model_list.append(D.score(X_test, Y_test))
predict=D.predict(X_test)
print('Accuracy:  ',accuracy_score(Y_test,predict))
print('Classification report: \n',classification_report(Y_test,predict))
print('Confusion matrix: \n',confusion_matrix(Y_test,predict))
```

```
Accuracy:   0.7181571815718157
Classification report:
              precision    recall  f1-score   support

           0       0.73      0.71      0.72      1681
           1       0.71      0.73      0.72      1640

    accuracy                           0.72      3321
   macro avg       0.72      0.72      0.72      3321
weighted avg       0.72      0.72      0.72      3321

Confusion matrix:
 [[1190  491]
 [ 445 1195]]
```

- Gaussian NB

```
NB = GaussianNB()
NB.fit(X_train,Y_train)
NB.score(X_test, Y_test)
model_list.append(NB.score(X_test, Y_test))
predict=NB.predict(X_test)
print('Accuracy:  ',accuracy_score(Y_test,predict))
print('Classification report: \n',classification_report(Y_test,predict))
print('Confusion matrix: \n',confusion_matrix(Y_test,predict))
```

```
Accuracy:   0.7398373983739838
Classification report:
              precision    recall  f1-score   support

           0       0.74      0.74      0.74      1681
           1       0.74      0.74      0.74      1640

    accuracy                           0.74      3321
   macro avg       0.74      0.74      0.74      3321
weighted avg       0.74      0.74      0.74      3321

Confusion matrix:
 [[1251  430]
 [ 434 1206]]
```

- SVC

```
S = SVC()
S.fit(X_train,Y_train)
S.score(X_test, Y_test)
model_list.append(S.score(X_test, Y_test))
predict=S.predict(X_test)
print('Accuracy:  ',accuracy_score(Y_test,predict))
print('Classification report: \n',classification_report(Y_test,predict))
print('Confusion matrix: \n',confusion_matrix(Y_test,predict))
```

```
Accuracy:   0.8732309545317676
Classification report:
              precision    recall  f1-score   support

           0       0.86      0.90      0.88      1681
           1       0.89      0.85      0.87      1640

    accuracy                           0.87      3321
   macro avg       0.87      0.87      0.87      3321
weighted avg       0.87      0.87      0.87      3321

Confusion matrix:
 [[1507  174]
 [ 247 1393]]
```

- Perceptron

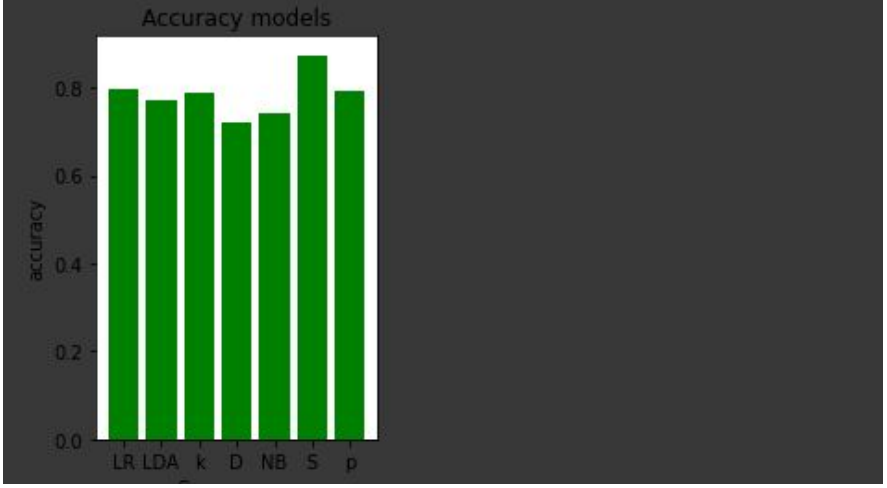
```
p = Perceptron()
p.fit(X_train,Y_train)
p.score(X_test, Y_test)
model_list.append(p.score(X_test, Y_test))
predict=p.predict(X_test)
print('Accuracy:  ',accuracy_score(Y_test,predict))
print('Classification report: \n',classification_report(Y_test,predict))
print('Confusion matrix: \n',confusion_matrix(Y_test,predict))
```

```
Accuracy:   0.7919301415236375
Classification report:
              precision    recall  f1-score   support

           0       0.76      0.86      0.81      1681
           1       0.83      0.73      0.78      1640

    accuracy                           0.79      3321
   macro avg       0.80      0.79      0.79      3321
weighted avg       0.80      0.79      0.79      3321

Confusion matrix:
 [[1439  242]
 [ 449 1191]]
```

So sánh độ chính xác của các model:

```
import matplotlib.pyplot as plt
plt.subplot(1,2,1)
names = ['LR','LDA','k','D','NB','S','p']
plt.bar(names,model_list,color='green', label =' accuracy')
plt.title(' Accuracy models ')
plt.xlabel('Compare acc')
plt.ylabel('accuracy')
```

```
Text(0, 0.5, 'accuracy')
```



Kết quả: chọn SVC làm model chính cho bài toán. Lưu lại model để tiếp tục sử dụng:

```
import pickle
# Save the trained model as a pickle string.
filename = '/content/drive/My Drive/Model/S.sav'
pickle.dump(SVC(), open(filename, 'wb'))
```

Bản chất model SVC là tìm ra siêu mặt phẳng phân lớp dữ liệu, với bài toán binary classification mà 2 classes là linearly separable thì có vô số siêu mặt phẳng phân biệt hai classes cho nên model này cho kết quả độ chính xác khá cao

Tinh chỉnh tham số của model SVC:

Dùng GridSearchCV tinh chỉnh các tham số ảnh hưởng đến độ chính xác của model. Ở SVC tinh 2 tham số là C và gamma. Training với bộ dữ liệu

```
from sklearn.model_selection import GridSearchCV
para = {'C':[1,3], 'gamma':['scale','auto']}
g= GridSearchCV(SVC(),param_grid= para)
g.fit(X_train,Y_train)
```

Chọn ra bộ tham số có độ chính xác cao nhất:

```
g.best_estimator_
```

```
SVC(C=3, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
g.score(X_test, Y_test)
```

```
0.8849744052996086
```

Bộ C =3 , gamma = scale cho kết quả 0.8849744052996086 so với model cũ là 0.8732309545317676 -> tham số mới cho kết quả tốt hơn, lưu lại model

```
import pickle
# Save the trained model as a pickle string.
filename2 = '/content/drive/My Drive/Model/newS.sav'
pickle.dump(newS, open(filename, 'wb'))
```

Tiếp tục tinh chỉnh tham số: giữ nguyên tham số cho gamma, thử C = 4,5 liệu kết quả có tốt hơn khi tăng C không:

```
para2 = {'C':[4,5], 'gamma':['scale']}
g2= GridSearchCV(SVC(),param_grid= para2)
g2.fit(X_train,Y_train)
```

Chọn ra bộ tham số có độ chính xác cao nhất:

```
g2.best_estimator_
```

```
SVC(C=5, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
g2.score(X_test, Y_test)
```

```
0.8846732911773563
```

Kết quả là 0.8846732911773563 thấp hơn so với 0.8849744052996086(tinh chỉnh lần 1) -> Dừng tinh chỉnh, chọn model tinh chỉnh lần 1 làm model chính thức.

Áp dụng model chính thức để nhận diện giới tính khuôn mặt từ 1 ảnh được nhập vào

Cài đặt các thư viện và hàm xử lí ảnh đầu vào trước khi đưa vào dự đoán: chuyển ảnh thành ảnh xám, resize:

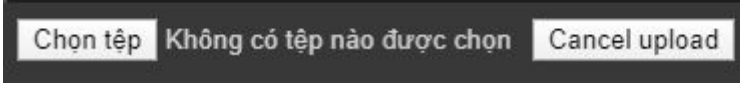
```
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix , precision_score, recall_score
import cv2
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import load_model
def detect (img):
  img_1 = img
  input = img
  ##cv2_imshow(img_1)
  input = cv2.cvtColor(input, cv2.COLOR_RGB2GRAY)
  input = cv2.resize(input,(50,50), Image.ANTIALIAS)
  input = np.reshape(input, 50*50)
  input = (np.asarray(input)/255.0)
  result = loaded_model.predict([input])
  return(result)
```

Cài đặt thư viện Face recognition

```
!pip install dlib
!pip install face_recognition
import face_recognition
```

Upload ảnh để nhận diện

```
from google.colab import files
uploaded = files.upload()
```



Chạy dự đoán gồm:

```
for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(name=fn, length=len(uploaded[fn])))

  image = face_recognition.load_image_file(fn)

  plt.imshow(image)
  plt.show()

  face_locations = face_recognition.face_locations(image)

  print("Found {} face(s) in this photograph.".format(len(face_locations)))

  for face_location in face_locations:
    top, right, bottom, left = face_location
    print("A face is located at pixel location Top: {}, Left: {}, Bottom: {}, Right: {}".format(top, left, bottom, right))
    face_image = image[top:bottom, left:right]
    plt.imshow(face_image)
    plt.show()
    if detect(face_image):
      print( "               Female")
    else:
      print("               Male")
```

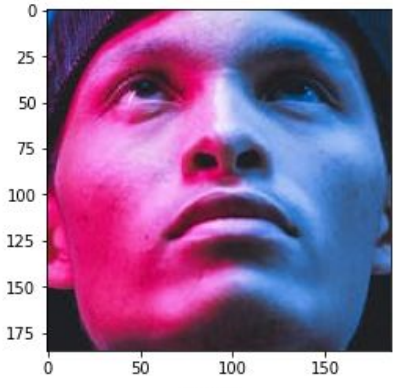
Kết quả nhận được:

```
Found 2 face(s) in this photograph.
```

```
A face is located at pixel location Top: 2
```

```
Female
```

```
A face is located at pixel location Top: 160,
```



```
Male
```

3. DEMO

Video demo được upload ở youtobe.

Link youtobe: https://www.youtube.com/watch?v=VAxq7OWbgYg

Hướng dẫn sử dụng model để chạy demo:

https://github.com/ToanToan110/Project_ML/blob/master/README.md

4. KẾT LUẬN

Tuy chọn được độ chính xác cao nhất trong các model đã traning nhưng dự đoán sai khá nhiều. Có thể do 1 số nguyên nhân sau:

- Ảnh chưa được qua nhiều xử lí
- Phương pháp rút trích đặc trưng còn đơn giản
- Thuật toán Face regconition chưa tối ưu: cùng 1 hình ảnh(ảnh chân dung riêng và ảnh được ghép với nhiều ảnh khác) nhưng cho 2 kết quả khác nhau (trong demo)
- Độ chính xác khi có nhiều khuôn mặt (từ 4 trở lên) giảm so với chỉ 1-2 khuôn mặt trong 1 hình

Hướng khắc phục và cải thiện

- Chọn ảnh chất lượng và tiền xử lí ảnh tốt hơn
- Tìm kiếm thuật toán nhận diện khuôn mặt khác

LINK THAO KHẢO

Nguồn dataset:

- Kaggle: https://www.kaggle.com/gmlmrinalini/genderdetectionface
- Viblo.asia:(https://viblo.asia/p/vn-celeb-du-lieu-khuon-mat-nguoi-noi-tieng-viet-nam-va-bai-toan-face-recognition-Az45bG9VKxY) : Dữ liệu được thu thập và chia sẻ công khai, bộ dữ liệu tiếp tục được mở rộng. Bộ dữ liệu được sử dụng trong cuộc thi "Nhận diện người nổi tiếng" (https://www.aivivn.com/contests/2)

Tham khảo code:

Code trong bài được tham khảo từ: https://www.kaggle.com/mervemutlu/classification-using-logistic-regression-beginner

Face recognition : https://pypi.org/project/face-recognition/

Link github chứa đồ án: https://github.com/ToanToan110/Project_ML

BẢNG PHÂN CÔNG TRÁCH NHIỆM THÀNH VIÊN

Tên thành viên	Trần Quốc Toàn	Nguyễn Tri Đức Anh	Trần Văn Thục
Công việc	● Soạn và chạy code ● Làm ứng dụng nhận diện	● Tìm dataset ● Báo cáo và quay demo	● _Làm slide và báo cáo