In [1]:
```
#importing all the required packages
import numpy as np
import pandas as pd
import math
import cv2
import matplotlib.pyplot as plt
import os
import seaborn as sns

from PIL import Image
from os import listdir
from os.path import isfile, join
from random import shuffle

import tensorflow as tf
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.utils.np_utils import to_categorical
```

In [2]:
```
#specifying dataset directory

os.chdir('D:\\6th sem\\Gender & Age classification\\utkface_aligned_cropped\\UTKFa
```

In [3]:
```
im =Image.open('25_1_2_20170104015850244.jpg.chip.jpg').resize((64,64))
im
```

Out[3]:

In [4]:
```
onlyfiles = os.listdir()
```

In [5]:
```
#counting number of image files in onlyfiles

len(onlyfiles)
```

Out[5]:
```
23151
```

In [6]:
```
#shuffling all the images in onlyfiles

shuffle(onlyfiles)
```

In [7]:
```
#getting age from the name of the image

age = [i.split('_')[0] for i in onlyfiles]
```

In [8]:
```
df = pd.DataFrame()
df['onlyfiles'], df['age'] = onlyfiles, age
```

In [9]:
```
#displaying top files present in dataframe using head

df.head()
```

Out[9]:

	onlyfiles	age
0	65_1_0_20170110183018880.jpg.chip.jpg	65
1	70_1_0_20170110122250867.jpg.chip.jpg	70
2	45_1_1_20170116003204136.jpg.chip.jpg	45
3	36_0_0_20170116182401789.jpg.chip.jpg	36
4	8_1_0_20170109202804967.jpg.chip.jpg	8

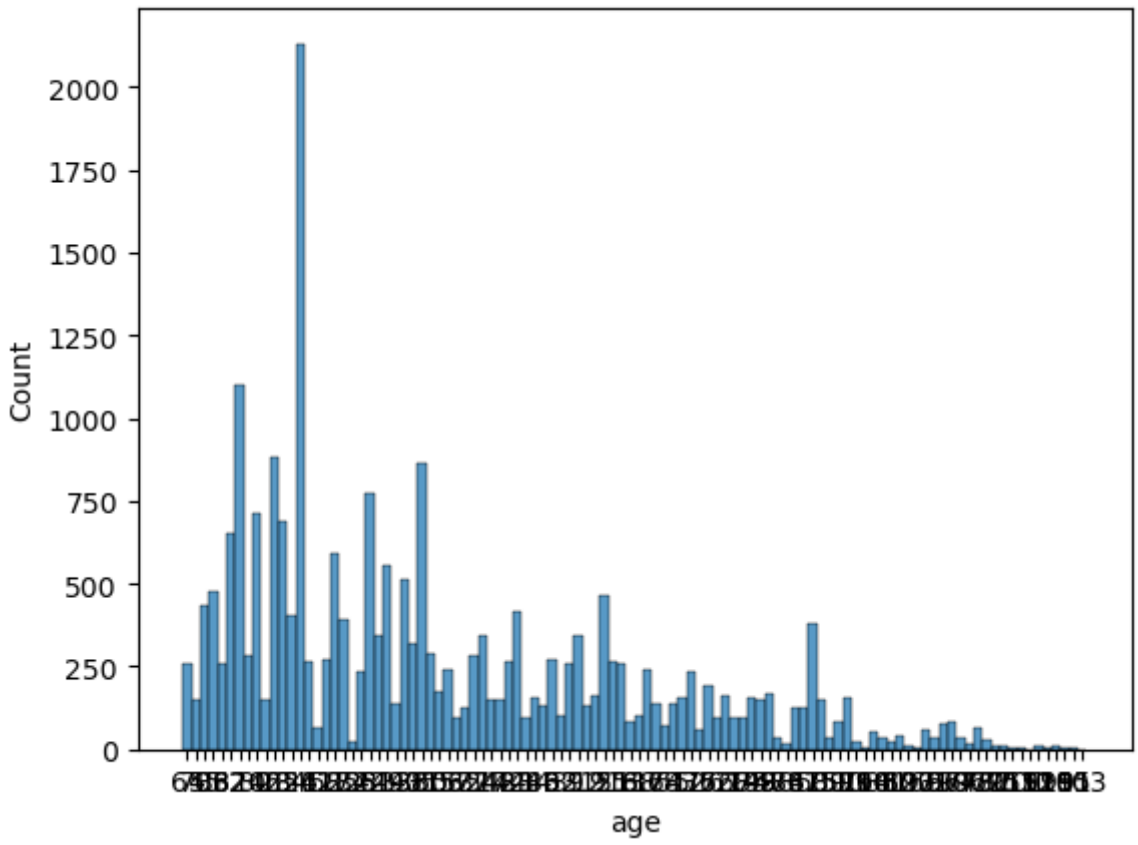
In [10]:
```
#displaying bottom files present in dataframe using tail

df.tail()
```

Out[10]:

	onlyfiles	age
23146	32_0_2_20170116182724023.jpg.chip.jpg	32
23147	65_1_3_20170109141910621.jpg.chip.jpg	65
23148	45_1_0_20170103183543298.jpg.chip.jpg	45
23149	26_1_1_20170112205853539.jpg.chip.jpg	26
23150	49_1_3_20170109132625049.jpg.chip.jpg	49

In [11]:
```
#displaying the no of image corresponing with the age

sns.histplot(df['age'])
```

Out[11]:
```
<Axes: xlabel='age', ylabel='Count'>
```



In [12]:
```
classes = []

#defining the age groups
```

```
for i in age:
    i = int(i)
    if i <= 5:                          #babies (1-5) CLASS 0
        classes.append(0)

    if (i>5) and (i<=12):               #Children (6-12) CLASS 1
        classes.append(1)

    if (i>12) and (i<=18):               #Youth (12-18) CLASS 2
        classes.append(2)

    if (i>18) and (i<40):                #ADULTS (19-40) CLASS 3
        classes.append(3)

    if (i>=40) and (i<60):               #MIDDLE AGED (40-60) 4
        classes.append(4)

    if i>=60:                            #Very Old (>60) CLASS 5
        classes.append(5)
```

In [13]:
```
# CONVERT IMAGES TO VECTORS
```

In [14]:
```
X_data =[]

#inserting each image one by one on X_data

for file in onlyfiles:
    face = cv2.imread(file)
    face =cv2.resize(face, (64, 64) )
    X_data.append(face)
```

In [15]:
```
# By applying np.squeeze(), the singleton dimensions were removed    (1, 3, 1) ->
#The main purpose of np.squeeze() is to reduce the dimensions of an array by remov

X = np.squeeze(X_data)
# X = np.array(X_data)
```

In [16]:
```
X.shape
```

Out[16]:
```
(23151, 64, 64, 3)
```

In [17]:
```
# normalizing data (0-1)

X = X.astype('float64')
X /= 255
```

In [18]:
```
classes[:10]
```

Out[18]:
```
[5, 5, 4, 3, 1, 3, 0, 3, 3, 4]
```

In [19]:
```
categorical_labels = to_categorical(classes, num_classes=6)
```

In [20]:
```
categorical_labels[:10]
```

Out[20]:
```
array([[0., 0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0.],
       [1., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0.]], dtype=float32)
```

In [22]:
```
from sklearn.model_selection import train_test_split
# Split the dataset for training, testing, and validation
x_train, x_test, y_train, y_test = train_test_split(X, categorical_labels, test_si
x_train, x_valid, y_train, y_valid = train_test_split(x_train, y_train, test_size=
```

In [24]:
```
from keras.preprocessing.image import ImageDataGenerator

# Data augmentation

datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

In [26]:
```
# Build the CNN model

model = Sequential()
model.add(Conv2D(filters=128, kernel_size=3, padding='same', activation='relu', in
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.25))

model.add(Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.25))

model.add(Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(6, activation='softmax'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 64, 64, 128)       3584

 max_pooling2d_3 (MaxPooling  (None, 32, 32, 128)      0
 2D)

 dropout_4 (Dropout)         (None, 32, 32, 128)       0

 conv2d_4 (Conv2D)           (None, 32, 32, 64)        73792

 max_pooling2d_4 (MaxPooling  (None, 16, 16, 64)       0
 2D)

 dropout_5 (Dropout)         (None, 16, 16, 64)        0

 conv2d_5 (Conv2D)           (None, 16, 16, 32)        18464

 max_pooling2d_5 (MaxPooling  (None, 8, 8, 32)         0
 2D)

 dropout_6 (Dropout)         (None, 8, 8, 32)          0

 flatten_1 (Flatten)         (None, 2048)              0

 dense_2 (Dense)             (None, 512)               1049088

 dropout_7 (Dropout)         (None, 512)               0

 dense_3 (Dense)             (None, 6)                 3078

=================================================================
Total params: 1,148,006
Trainable params: 1,148,006
Non-trainable params: 0
_________________________________________________________________
```

In [30]:
```
#compiling model

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [31]:
```
# Fit the model with data augmentation
history = model.fit(datagen.flow(x_train, y_train, batch_size=64),
                    steps_per_epoch=len(x_train) // 64,
                    epochs=25,
                    validation_data=(x_valid, y_valid))
```

```
Epoch 1/30
251/251 [==============================] - 185s 725ms/step - loss: 1.3028 - accura
cy: 0.5356 - val_loss: 1.1135 - val_accuracy: 0.6232
Epoch 2/30
251/251 [==============================] - 183s 731ms/step - loss: 1.0088 - accura
cy: 0.6282 - val_loss: 0.9303 - val_accuracy: 0.6680
Epoch 3/30
251/251 [==============================] - 183s 731ms/step - loss: 0.9292 - accura
cy: 0.6522 - val_loss: 0.8459 - val_accuracy: 0.6858
Epoch 4/30
251/251 [==============================] - 183s 731ms/step - loss: 0.8741 - accura
cy: 0.6709 - val_loss: 0.8013 - val_accuracy: 0.6977
Epoch 5/30
251/251 [==============================] - 183s 731ms/step - loss: 0.8319 - accura
cy: 0.6863 - val_loss: 0.7776 - val_accuracy: 0.7043
Epoch 6/30
251/251 [==============================] - 182s 726ms/step - loss: 0.8046 - accura
cy: 0.6918 - val_loss: 0.7284 - val_accuracy: 0.7233
Epoch 7/30
251/251 [==============================] - 183s 730ms/step - loss: 0.7797 - accura
cy: 0.6990 - val_loss: 0.7103 - val_accuracy: 0.7215
Epoch 8/30
251/251 [==============================] - 89s 354ms/step - loss: 0.7540 - accurac
y: 0.7067 - val_loss: 0.7249 - val_accuracy: 0.7283
Epoch 9/30
251/251 [==============================] - 65s 258ms/step - loss: 0.7326 - accurac
y: 0.7157 - val_loss: 0.6834 - val_accuracy: 0.7290
Epoch 10/30
251/251 [==============================] - 65s 259ms/step - loss: 0.7151 - accurac
y: 0.7205 - val_loss: 0.6787 - val_accuracy: 0.7327
Epoch 11/30
251/251 [==============================] - 66s 264ms/step - loss: 0.7000 - accurac
y: 0.7237 - val_loss: 0.6826 - val_accuracy: 0.7313
Epoch 12/30
251/251 [==============================] - 66s 263ms/step - loss: 0.6920 - accurac
y: 0.7295 - val_loss: 0.7028 - val_accuracy: 0.7155
Epoch 13/30
251/251 [==============================] - 65s 259ms/step - loss: 0.6702 - accurac
y: 0.7364 - val_loss: 0.6748 - val_accuracy: 0.7473
Epoch 14/30
251/251 [==============================] - 68s 270ms/step - loss: 0.6533 - accurac
y: 0.7413 - val_loss: 0.6659 - val_accuracy: 0.7402
Epoch 15/30
251/251 [==============================] - 67s 269ms/step - loss: 0.6485 - accurac
y: 0.7408 - val_loss: 0.6595 - val_accuracy: 0.7407
Epoch 16/30
251/251 [==============================] - 66s 261ms/step - loss: 0.6382 - accurac
y: 0.7468 - val_loss: 0.6585 - val_accuracy: 0.7377
Epoch 17/30
251/251 [==============================] - 68s 271ms/step - loss: 0.6248 - accurac
y: 0.7489 - val_loss: 0.6444 - val_accuracy: 0.7500
Epoch 18/30
251/251 [==============================] - 66s 262ms/step - loss: 0.6208 - accurac
y: 0.7538 - val_loss: 0.6529 - val_accuracy: 0.7452
Epoch 19/30
251/251 [==============================] - 66s 262ms/step - loss: 0.6019 - accurac
y: 0.7575 - val_loss: 0.6513 - val_accuracy: 0.7460
Epoch 20/30
251/251 [==============================] - 67s 266ms/step - loss: 0.5945 - accurac
y: 0.7602 - val_loss: 0.6524 - val_accuracy: 0.7470
Epoch 21/30
251/251 [==============================] - 69s 274ms/step - loss: 0.5886 - accurac
y: 0.7650 - val_loss: 0.6520 - val_accuracy: 0.7497
Epoch 22/30
```

```
251/251 [==============================] - 68s 270ms/step - loss: 0.5799 - accurac
y: 0.7650 - val_loss: 0.6443 - val_accuracy: 0.7490
Epoch 23/30
251/251 [==============================] - 73s 293ms/step - loss: 0.5685 - accurac
y: 0.7694 - val_loss: 0.6567 - val_accuracy: 0.7472
Epoch 24/30
251/251 [==============================] - 76s 303ms/step - loss: 0.5624 - accurac
y: 0.7710 - val_loss: 0.6750 - val_accuracy: 0.7455
Epoch 25/30
251/251 [==============================] - 75s 300ms/step - loss: 0.5623 - accurac
y: 0.7749 - val_loss: 0.6701 - val_accuracy: 0.7443
Epoch 26/30
251/251 [==============================] - 70s 280ms/step - loss: 0.5465 - accurac
y: 0.7752 - val_loss: 0.6671 - val_accuracy: 0.7452
Epoch 27/30
251/251 [==============================] - 70s 279ms/step - loss: 0.5325 - accurac
y: 0.7810 - val_loss: 0.6452 - val_accuracy: 0.7520
Epoch 28/30
251/251 [==============================] - 68s 271ms/step - loss: 0.5280 - accurac
y: 0.7865 - val_loss: 0.6612 - val_accuracy: 0.7480
Epoch 29/30
251/251 [==============================] - 67s 266ms/step - loss: 0.5191 - accurac
y: 0.7845 - val_loss: 0.6536 - val_accuracy: 0.7543
Epoch 30/30
251/251 [==============================] - 66s 262ms/step - loss: 0.5163 - accurac
y: 0.7871 - val_loss: 0.6701 - val_accuracy: 0.7440
```

Out[31]:
```
<keras.callbacks.History at 0x1b7a22732b0>
```

In [32]:
```
# Evaluating the model on test set
score = model.evaluate(x_test, y_test)

#test accuracy
print('\n', 'Test accuracy:', score[1])
```

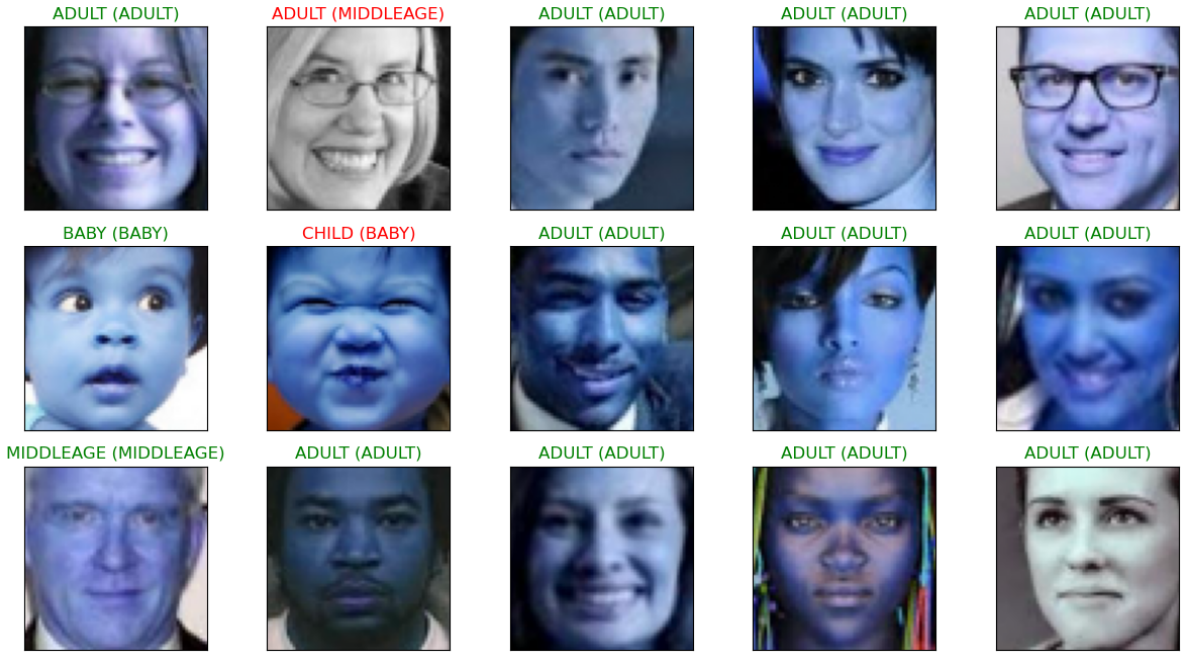
```
48/48 [==============================] - 1s 29ms/step - loss: 0.6635 - accuracy:
0.7531

 Test accuracy: 0.7531106472015381
```

In [33]:
```
#labeling  age groups

labels =["BABY",         # index 0
        "CHILD",        # index 1
        "YOUTH",        # index 2
        "ADULT",        # index 3
        "MIDDLEAGE",    # index 4
        "OLD",          # index 5
        ]
```

In [40]:
```
y_hat = model.predict(x_test)

# Plot a random sample of 15 test images, their predicted labels and ground truth

figure = plt.figure(figsize=(15, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks=[])

    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    predict_index = np.argmax(y_hat[index])
    true_index = np.argmax(y_test[index])

    # Set the title for each image
```

```
    ax.set_title("{} ({})".format(labels[predict_index],
                                  labels[true_index]),
                                  color=("green" if predict_index == true_index el
plt.show()
```

```
48/48 [==============================] - 1s 28ms/step
```



In [41]:
```
model.save('D:\\6th sem\\Gender & Age classification\\Age_Classification_acc75_ep3
```

In [ ]:
```
#Loading model and Detecting images from model
```

In [1]:
```
from tensorflow.keras.models import load_model
from sklearn.metrics import accuracy_score
```

In [5]:
```
model=load_model('D:\\6th sem\\Gender & Age classification\\Age_Classification_acc
```

In [2]:
```
# x_test[0]
```

In [61]:
```
import cv2
import numpy as np
```

In [62]:
```
image=cv2.imshow('Age',x_test[0])
cv2.waitKey(0)
```

Out[62]:
```
-1
```

In [66]:
```
img_path=input("Enter the image path: ")
img=cv2.imread(img_path)

cv2.imshow('Age',img)
cv2.waitKey(0)
cv2.destroyAllWindows()

img=np.array(img)
img=cv2.resize(img,(64,64))
img.shape
img=np.expand_dims(img,0)

output=model.predict(img)
final_out=np.argmax(output)
print("The given image is: ",labels[final_out])
```

```
Enter the image path: D:\6th sem\Gender & Age classification\Gender_Dataset\man\fa
ce_1075.jpg
1/1 [==============================] - 0s 9ms/step
The given image is:  ADULT
```

In [ ]:
```
face_1075
```

In [ ]:
```

```