ISTANBUL TECHNICAL UNIVERSITY, COMPUTER ENGINEERING

BLG521E Artificial Intelligence Project Progress Report

Anıl Öztürk

504181504

December 6, 2019

1 THE PROJECT DEFINITION

The aim of the project is to develop **a concept software that provides the most compatible advertisement according to the population distribution in front of a billboard**. The software tries to show the advertisements of the companies whose target is closest to the current audience by extracting the age and gender statistics of the audience comes from the camera feed on top of the billboard. The algorithm will be running on **Python3** and models will be created, trained and executed on **Keras** or **PyTorch**. **State-of-art object detectors** and **own estimators** will be used.

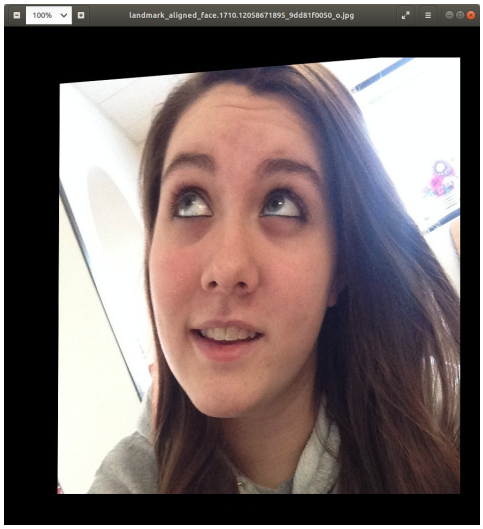
2 THE PROGRESS

2.1 PREPARING THE DATASETS

For the project, I have downloaded two different datasets with different layout from different open-source platforms. Firstly, I needed to preprocess the data and put them together in a common scheme.

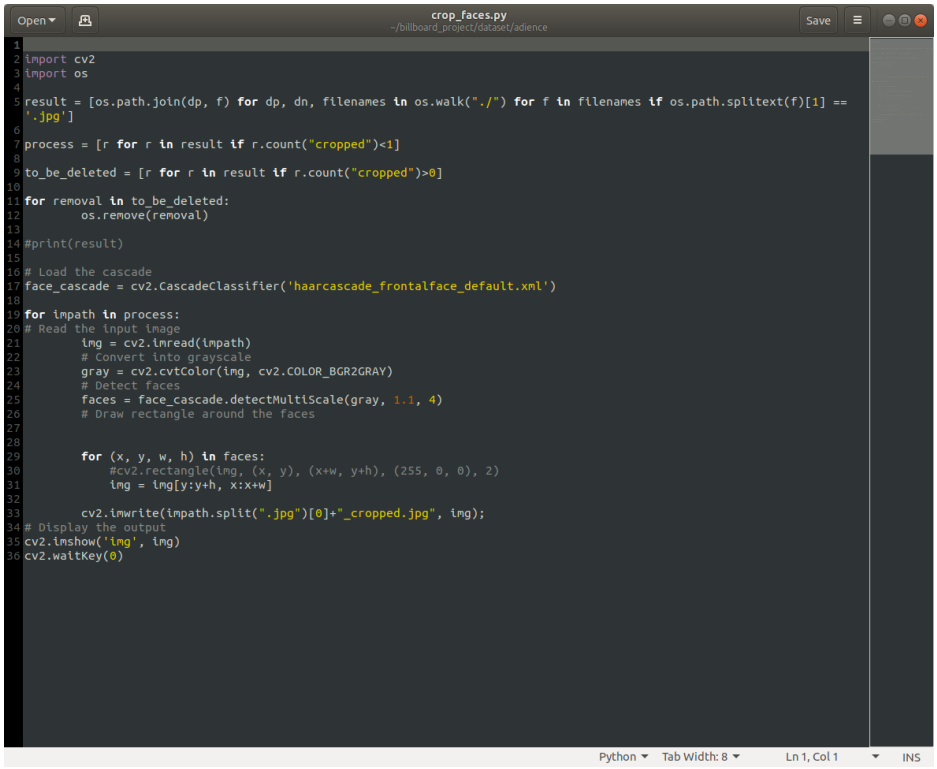
2.1.1 ADIENCE DATASET

Adience dataset includes **about 19300** images of human faces from different age intervals and genders. But it was in need for little pre-processing. An example image from dataset is below;

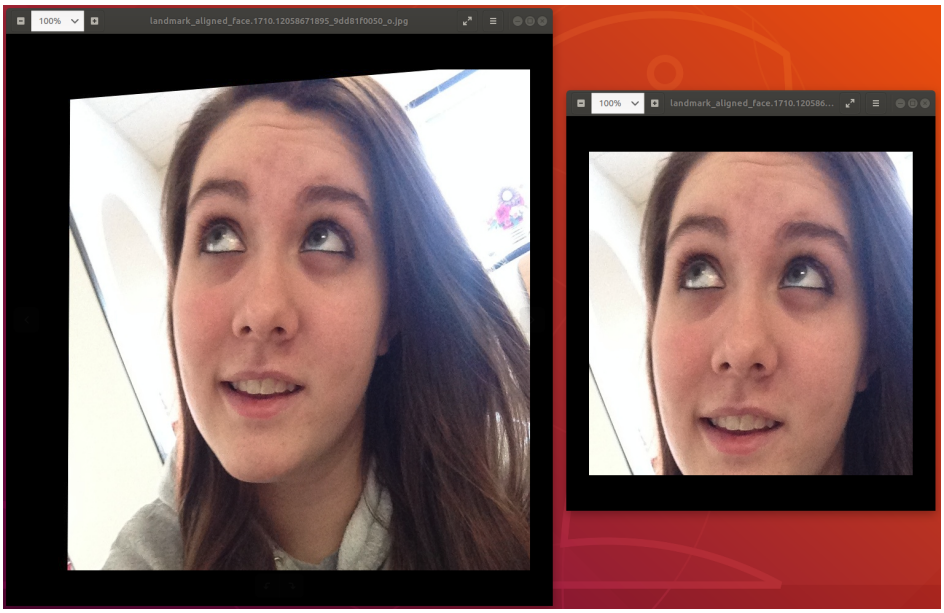


The faces in images need to be cropped since we will run a face detector in our project in the first place. We will get face-cropped images. If we train our gender-age predictor with these uncropped images, we will run into stability issues.

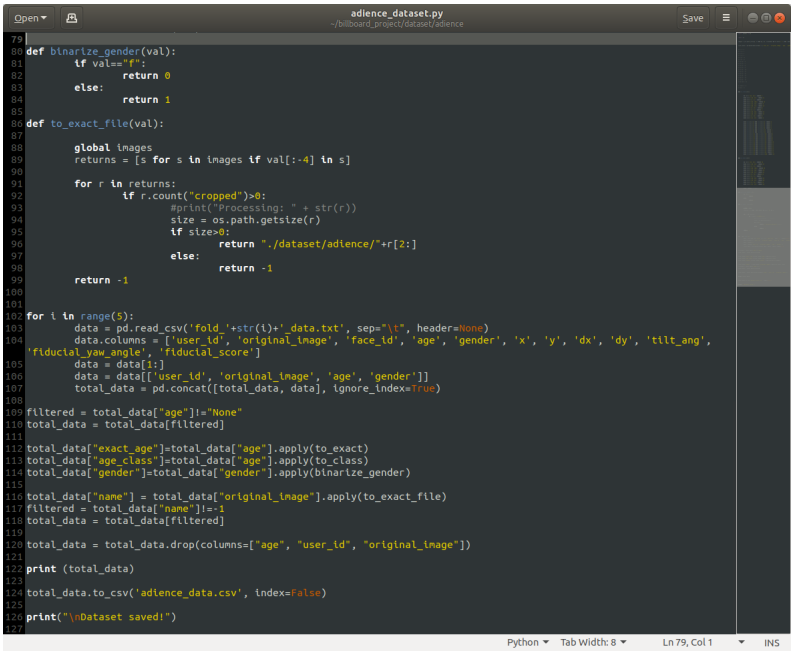
I wrote a simple cropper script that crops faces and saves the cropped image with a new name. The script uses **Haar-Cascade** detector since most of the images are frontal-face images as I mentioned before in the **RAD Report**.



The result is below;

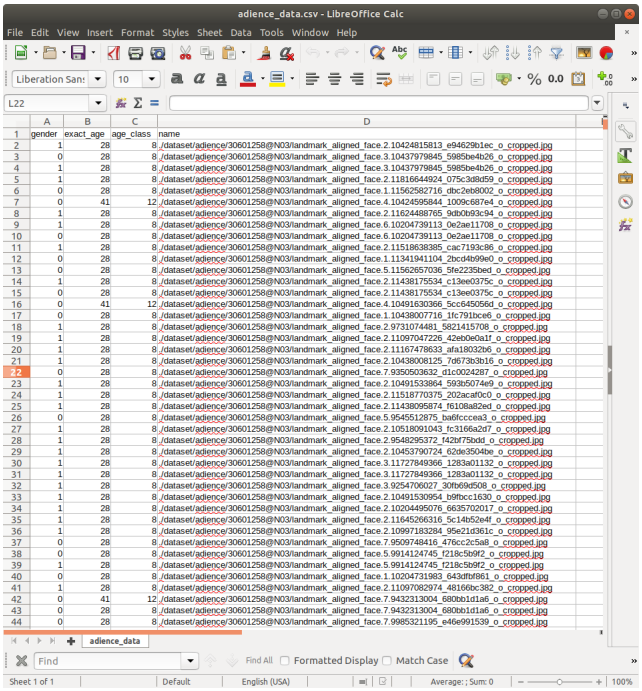


The dataset has mixed age labels. Some of the images have intervals like **(8-15), (25-32)**, but some of them have exact age labels like **27, 41, 63**... So I wrote a script that tidies this situation up. Saves the whole data into a **.csv** with exact values and age group IDs. For the data with only interval information, I assumed the **mean** of the interval as the exact value.




```
def binarize_gender(val):
        if val=="f":
                return 0
        else:
                return 1

def to_exact_file(val):

        global images
        returns = [s for s in images if val[:-4] in s]

        for r in returns:
                if r.count("cropped")>0:
                        #print("Processing: " + str(r))
                        size = os.path.getsize(r)
                        if size>0:
                                return "./dataset/adience/"+r[2:]
                        else:
                                return -1
        return -1


for i in range(5):
        data = pd.read_csv('fold_'+str(i)+'_data.txt', sep="\t", header=None)
        data.columns = ['user_id', 'original_image', 'face_id', 'age', 'gender', 'x', 'y', 'dx', 'dy', 'tilt_ang',
'fiducial_yaw_angle', 'fiducial_score']
        data = data[1:]
        data = data[['user_id', 'original_image', 'age', 'gender']]
        total_data = pd.concat([total_data, data], ignore_index=True)

filtered = total_data["age"]!="None"
total_data = total_data[filtered]

total_data["exact_age"]=total_data["age"].apply(to_exact)
total_data["age_class"]=total_data["age"].apply(to_class)
total_data["gender"]=total_data["gender"].apply(binarize_gender)

total_data["name"] = total_data["original_image"].apply(to_exact_file)
filtered = total_data["name"]!=-1
total_data = total_data[filtered]

total_data = total_data.drop(columns=["age", "user_id", "original_image"])

print (total_data)

total_data.to_csv('adience_data.csv', index=False)

print("\nDataset saved!")
```

Here is the dataset I've created from the Adience data;



	A	B	C	D
1	gender	exact_age	age_class	name
2	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.10424815813_e94629b1ec_o_cropped.jpg
3	0	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.3.10437979845_5985be4b26_o_cropped.jpg
4	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.3.10437979845_5985be4b26_o_cropped.jpg
5	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.11816644924_075c3d8d59_o_cropped.jpg
6	0	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.1.11562582716_dbc2eb8002_o_cropped.jpg
7	0	41	12	./dataset/adience/30601258@N03/landmark_aligned_face.4.10424595844_1009c687e4_o_cropped.jpg
8	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.11624488765_9db0b93c94_o_cropped.jpg
9	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.6.10204739113_0e2ae11708_o_cropped.jpg
10	0	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.6.10204739113_0e2ae11708_o_cropped.jpg
11	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.11518638385_cac7193c86_o_cropped.jpg
12	0	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.1.11341941104_2bcd4b99e0_o_cropped.jpg
13	0	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.5.11562657036_5fe2235bed_o_cropped.jpg
14	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.11438175534_c13ee0375c_o_cropped.jpg
15	0	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.11438175534_c13ee0375c_o_cropped.jpg
16	0	41	12	./dataset/adience/30601258@N03/landmark_aligned_face.4.10491630396_5cc645056d_o_cropped.jpg
17	0	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.1.10438007716_1fc791bce6_o_cropped.jpg
18	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.9731074481_5821415708_o_cropped.jpg
19	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.11097047226_42eb0e0a1f_o_cropped.jpg
20	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.11167478633_afa18032b6_o_cropped.jpg
21	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.10438008125_7d673b3b16_o_cropped.jpg
22	0	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.7.9350503632_d1c0024287_o_cropped.jpg
23	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.10491533864_593b5074e9_o_cropped.jpg
24	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.11518770375_202acaf0c0_o_cropped.jpg
25	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.11438095874_f6108a82ed_o_cropped.jpg
26	0	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.5.9545512875_ba6fcccea3_o_cropped.jpg
27	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.10518091043_fc3166a2d7_o_cropped.jpg
28	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.9548295372_f42bf75bdd_o_cropped.jpg
29	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.10453790724_62de3504be_o_cropped.jpg
30	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.3.11727849366_1283a01132_o_cropped.jpg
31	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.3.11727849366_1283a01132_o_cropped.jpg
32	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.3.9254706027_30fb69d508_o_cropped.jpg
33	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.10491530954_b9bcc1630_o_cropped.jpg
34	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.10204495076_6635702017_o_cropped.jpg
35	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.11645266316_5c14b52e4f_o_cropped.jpg
36	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.10997183284_95e21d361c_o_cropped.jpg
37	0	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.7.9509748416_476cc2c5a8_o_cropped.jpg
38	0	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.5.9914124745_f218c5b9f2_o_cropped.jpg
39	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.5.9914124745_f218c5b9f2_o_cropped.jpg
40	0	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.1.10204731983_643dfbf861_o_cropped.jpg
41	1	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.2.11097082974_48166bc382_o_cropped.jpg
42	0	41	12	./dataset/adience/30601258@N03/landmark_aligned_face.7.9432313004_680bb1d1a6_o_cropped.jpg
43	0	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.7.9432313004_680bb1d1a6_o_cropped.jpg
44	0	28	8	./dataset/adience/30601258@N03/landmark_aligned_face.7.9985321195_e46e991539_o_cropped.jpg

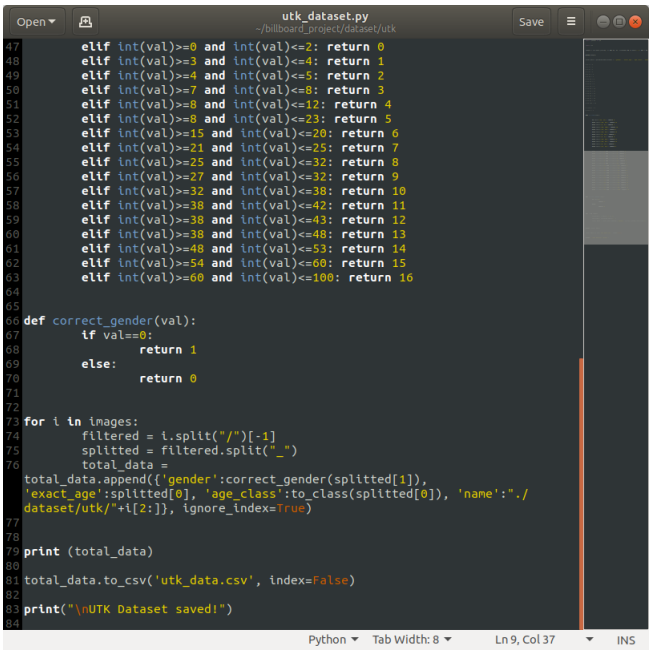
2.1.2 UTKFace Dataset

UTKFace dataset includes **about 21000** images of human faces from different age intervals and genders. Samples from dataset is below;

The information about the images are stored in images' name like;

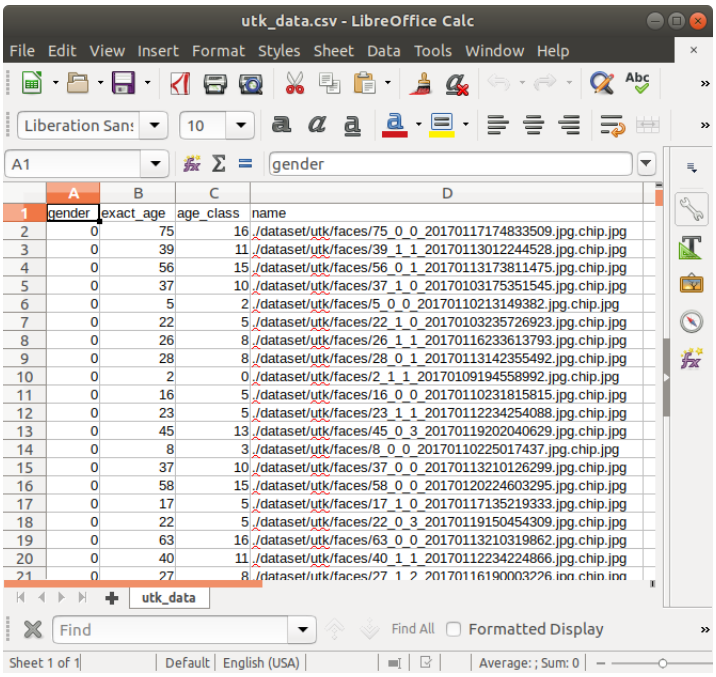
[age]_[gender]_[race]_[date&time].jpg

I iterated over the images and created a dataframe .csv file from their information. The script I wrote and used is below;



```
        elif int(val)>=0 and int(val)<=2: return 0
        elif int(val)>=3 and int(val)<=4: return 1
        elif int(val)>=4 and int(val)<=5: return 2
        elif int(val)>=7 and int(val)<=8: return 3
        elif int(val)>=8 and int(val)<=12: return 4
        elif int(val)>=8 and int(val)<=23: return 5
        elif int(val)>=15 and int(val)<=20: return 6
        elif int(val)>=21 and int(val)<=25: return 7
        elif int(val)>=25 and int(val)<=32: return 8
        elif int(val)>=27 and int(val)<=32: return 9
        elif int(val)>=32 and int(val)<=38: return 10
        elif int(val)>=38 and int(val)<=42: return 11
        elif int(val)>=38 and int(val)<=43: return 12
        elif int(val)>=38 and int(val)<=48: return 13
        elif int(val)>=48 and int(val)<=53: return 14
        elif int(val)>=54 and int(val)<=60: return 15
        elif int(val)>=60 and int(val)<=100: return 16


def correct_gender(val):
        if val==0:
                return 1
        else:
                return 0


for i in images:
        filtered = i.split("/")[-1]
        splitted = filtered.split("_")
        total_data =
total_data.append({'gender':correct_gender(splitted[1]),
'exact_age':splitted[0], 'age_class':to_class(splitted[0]), 'name':"./
dataset/utk/"+i[2:]}, ignore_index=True)


print (total_data)

total_data.to_csv('utk_data.csv', index=False)

print("\nUTK Dataset saved!")
```

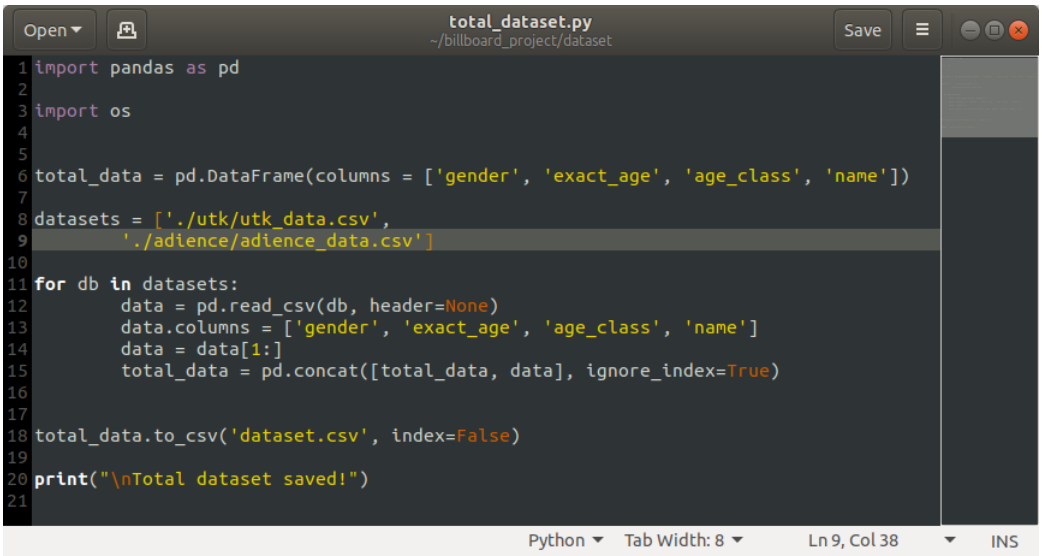
The dataset extracted from the images is below;



	A	B	C	D
1	gender	exact_age	age_class	name
2	0	75	16	./dataset/utk/faces/75_0_0_20170117174833509.jpg.chip.jpg
3	0	39	11	./dataset/utk/faces/39_1_1_20170113012244528.jpg.chip.jpg
4	0	56	15	./dataset/utk/faces/56_0_1_20170113173811475.jpg.chip.jpg
5	0	37	10	./dataset/utk/faces/37_1_0_20170103175351545.jpg.chip.jpg
6	0	5	2	./dataset/utk/faces/5_0_0_20170110213149382.jpg.chip.jpg
7	0	22	5	./dataset/utk/faces/22_1_0_20170103235726923.jpg.chip.jpg
8	0	26	8	./dataset/utk/faces/26_1_1_20170116233613793.jpg.chip.jpg
9	0	28	8	./dataset/utk/faces/28_0_1_20170113142355492.jpg.chip.jpg
10	0	2	0	./dataset/utk/faces/2_1_1_20170109194558992.jpg.chip.jpg
11	0	16	5	./dataset/utk/faces/16_0_0_20170110231815815.jpg.chip.jpg
12	0	23	5	./dataset/utk/faces/23_1_1_20170112234254088.jpg.chip.jpg
13	0	45	13	./dataset/utk/faces/45_0_3_20170119202040629.jpg.chip.jpg
14	0	8	3	./dataset/utk/faces/8_0_0_20170110225017437.jpg.chip.jpg
15	0	37	10	./dataset/utk/faces/37_0_0_20170113210126299.jpg.chip.jpg
16	0	58	15	./dataset/utk/faces/58_0_0_20170120224603295.jpg.chip.jpg
17	0	17	5	./dataset/utk/faces/17_1_0_20170117135219333.jpg.chip.jpg
18	0	22	5	./dataset/utk/faces/22_0_3_20170119150454309.jpg.chip.jpg
19	0	63	16	./dataset/utk/faces/63_0_0_20170113210319862.jpg.chip.jpg
20	0	40	11	./dataset/utk/faces/40_1_1_20170112234224866.jpg.chip.jpg
21	0	27	8	./dataset/utk/faces/27_1_2_20170116190003226.jpg.chip.jpg

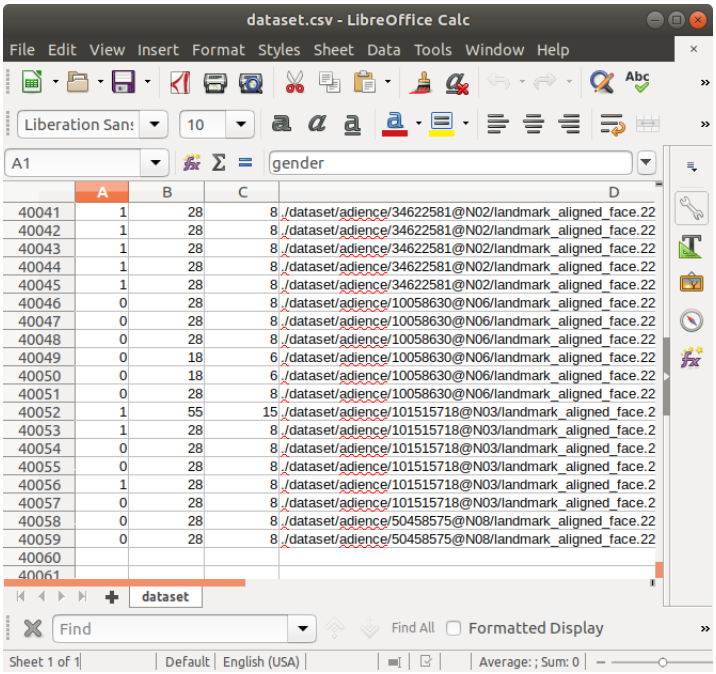
2.1.3 MERGING THE DATASETS

We prepared both datasets with one common structure. Now, we can merge them together. This is the script I wrote to merge two datasets;




```
import pandas as pd

import os


total_data = pd.DataFrame(columns = ['gender', 'exact_age', 'age_class', 'name'])

datasets = ['./utk/utk_data.csv',
        './adience/adience_data.csv']

for db in datasets:
        data = pd.read_csv(db, header=None)
        data.columns = ['gender', 'exact_age', 'age_class', 'name']
        data = data[1:]
        total_data = pd.concat([total_data, data], ignore_index=True)


total_data.to_csv('dataset.csv', index=False)

print("\nTotal dataset saved!")

```

This is the total database;



	A	B	C	D
40041	1	28	8	./dataset/adience/34622581@N02/landmark_aligned_face.22
40042	1	28	8	./dataset/adience/34622581@N02/landmark_aligned_face.22
40043	1	28	8	./dataset/adience/34622581@N02/landmark_aligned_face.22
40044	1	28	8	./dataset/adience/34622581@N02/landmark_aligned_face.22
40045	1	28	8	./dataset/adience/34622581@N02/landmark_aligned_face.22
40046	0	28	8	./dataset/adience/10058630@N06/landmark_aligned_face.22
40047	0	28	8	./dataset/adience/10058630@N06/landmark_aligned_face.22
40048	0	28	8	./dataset/adience/10058630@N06/landmark_aligned_face.22
40049	0	18	6	./dataset/adience/10058630@N06/landmark_aligned_face.22
40050	0	18	6	./dataset/adience/10058630@N06/landmark_aligned_face.22
40051	0	28	8	./dataset/adience/10058630@N06/landmark_aligned_face.22
40052	1	55	15	./dataset/adience/101515718@N03/landmark_aligned_face.2
40053	1	28	8	./dataset/adience/101515718@N03/landmark_aligned_face.2
40054	0	28	8	./dataset/adience/101515718@N03/landmark_aligned_face.2
40055	0	28	8	./dataset/adience/101515718@N03/landmark_aligned_face.2
40056	1	28	8	./dataset/adience/101515718@N03/landmark_aligned_face.2
40057	0	28	8	./dataset/adience/101515718@N03/landmark_aligned_face.2
40058	0	28	8	./dataset/adience/50458575@N08/landmark_aligned_face.22
40059	0	28	8	./dataset/adience/50458575@N08/landmark_aligned_face.22
40060				
40061				

Our data size is **40059** after all this pre-processing. It's a good value for training CNN models.

2.2 Training Phase

2.2.1 Creating Features and Labels

I decided to resize all faces to **32x32** and use age-groups for estimating ages.

Creating Dataset

I am creating the pandas dataframe from the database I have previously created. Then, I resize all the images to 32x32, for my neural network.

In [3]:
```
## CREATING THE DATASET

traindata = pd.read_csv("./dataset/dataset.csv", header=None)
traindata.columns = ['gender', 'exact_age', 'age_class', 'name']
traindata = traindata[1:]
x_train_names = traindata["name"].values

X = []

for i in tqdm(range(len(x_train_names)), "Loading - Resizing Images", leave=False):
    image = cv2.imread(x_train_names[i], cv2.IMREAD_COLOR)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = cv2.resize(image, (32,32) )
    X.append(image)

X = np.array(X)

print("Total Data Count: " +str(len(X)))
```

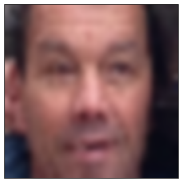
```
Total Data Count: 40058
```

In [4]:
```
## CREATING POSSIBLE LABEL ARRAYS

y_exact = [int(val) for val in traindata["exact_age"].values]
y_group = [int(val) for val in traindata["age_class"].values]
y_gender = [int(val) for val in traindata["gender"].values]

print("Samples from exact age labels: " + str(y_exact[:5]))
print("Samples from age group labels: " + str(y_group[:5]))
print("Samples from gender labels: " + str(y_gender[:5]))
```

```
Samples from exact age labels: [75, 39, 56, 37, 5]
Samples from age group labels: [16, 11, 15, 10, 2]
Samples from gender labels: [1, 0, 1, 0, 1]
```

I looked at some sample faces to check if I loaded and resized them correctly. To estimate age values as age groups, we must convert age-group id numbers to one-hot encoded vectors.

In [5]:
```
## GETTING A SAMPLE FROM THE DATASET

def see_img(img):
    plt.imshow(img, interpolation='bicubic')
    plt.xticks([]), plt.yticks([])
    plt.show()

see_img(X[2560])
see_img(X[32895])
```



In [22]:
```
## CREATING TRAIN, TEST, VALIDATION SPLITS

y_group_cat = np.asarray([to_categorical(i, num_classes=17) for i in y_group])
y_gender = np.asarray(y_gender)

X_train, X_test, y_gender_train, y_gender_test = train_test_split(X, y_gender, test_size=0.2, random_state=1)

X_train, X_test, y_group_train, y_group_test = train_test_split(X, y_group_cat, test_size=0.2, random_state=1)
```

The network I've created is below;

```
x = Conv2D(32, 7, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.6)(x)

x = Conv2D(32, 5, activation='relu')(x)
x = BatchNormalization()(x)
x = MaxPool2D(2)(x)
x = Dropout(0.6)(x)

x = Conv2D(64, 3, activation='relu')(x)
x = BatchNormalization()(x)
x = MaxPool2D(2)(x)
x = Dropout(0.6)(x)

x = Conv2D(128, 3, activation='relu')(x)
x = BatchNormalization()(x)
x = MaxPool2D(2)(x)
x = Dropout(0.6)(x)

x = Conv2D(256, 3, activation='relu')(x)
x = BatchNormalization()(x)
x = MaxPool2D(2)(x)
x = Dropout(0.6)(x)

x = Conv2D(256, 3, activation='relu')(x)
x = BatchNormalization()(x)
x = MaxPool2D(2)(x)
x = Dropout(0.6)(x)

x = Flatten()(x)

x1 = Dense(1024, activation='relu')(x)
x1 = BatchNormalization()(x1)
x1 = Dropout(0.5)(x1)

x1 = Dense(1024, activation='relu')(x1)
x1 = BatchNormalization()(x1)
x1 = Dropout(0.5)(x1)

x2 = Dense(1024, activation='relu')(x)
x2 = BatchNormalization()(x2)
x2 = Dropout(0.7)(x2)

x2 = Dense(1024, activation='relu')(x2)
x2 = BatchNormalization()(x2)
x2 = Dropout(0.7)(x2)

x1 = Dense(2, activation='softmax',name='sex_out')(x1)
x2 = Dense(11, activation='softmax',name='age_out')(x2)
```

It will change over time. This is the initial form of my model. I will try to feed age prediction with previous sex prediction.

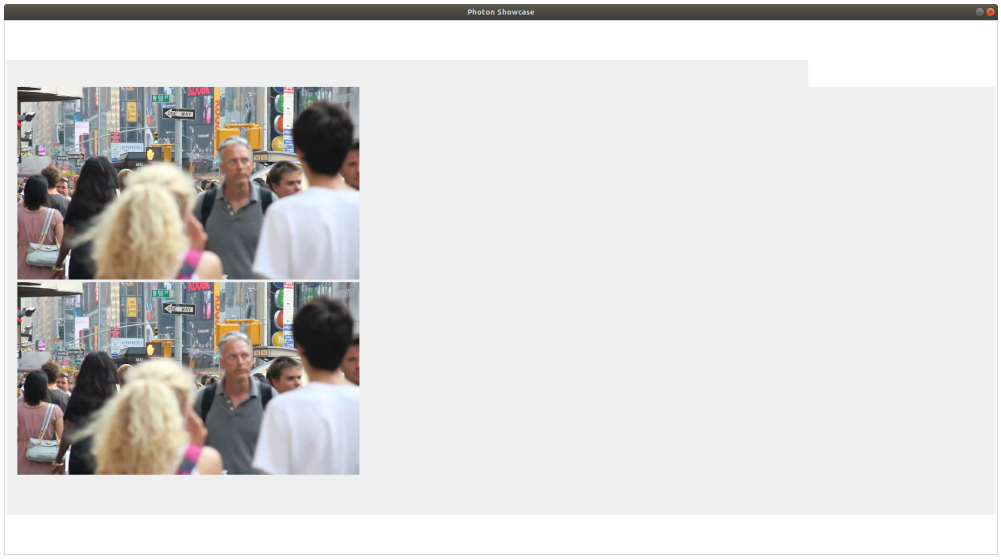
The current performance of my model is below;

```
Epoch 20/25
16128/16128 [==============================] - 20s 1ms/step - loss: 2.2882 - sex_out_loss: 0.6431 - age_out_loss: 1.6452
- sex_out_acc: 0.6370 - age_out_acc: 0.4420 - val_loss: 2.3739 - val_sex_out_loss: 0.6455 - val_age_out_loss: 1.7285 - v
al_sex_out_acc: 0.6300 - val_age_out_acc: 0.3544
Epoch 21/25
16128/16128 [==============================] - 20s 1ms/step - loss: 2.2726 - sex_out_loss: 0.6451 - age_out_loss: 1.6275
- sex_out_acc: 0.6430 - age_out_acc: 0.4467 - val_loss: 2.3426 - val_sex_out_loss: 0.6393 - val_age_out_loss: 1.7033 - v
al_sex_out_acc: 0.6289 - val_age_out_acc: 0.3454
Epoch 22/25
16128/16128 [==============================] - 20s 1ms/step - loss: 2.2542 - sex_out_loss: 0.6397 - age_out_loss: 1.6144
- sex_out_acc: 0.6444 - age_out_acc: 0.4549 - val_loss: 2.3375 - val_sex_out_loss: 0.6391 - val_age_out_loss: 1.6984 - v
al_sex_out_acc: 0.6462 - val_age_out_acc: 0.3672
Epoch 23/25
16128/16128 [==============================] - 20s 1ms/step - loss: 2.2501 - sex_out_loss: 0.6407 - age_out_loss: 1.6094
- sex_out_acc: 0.6461 - age_out_acc: 0.4557 - val_loss: 2.4173 - val_sex_out_loss: 0.6688 - val_age_out_loss: 1.7485 - v
al_sex_out_acc: 0.6261 - val_age_out_acc: 0.3633
Epoch 24/25
16128/16128 [==============================] - 20s 1ms/step - loss: 2.2467 - sex_out_loss: 0.6365 - age_out_loss: 1.6103
- sex_out_acc: 0.6490 - age_out_acc: 0.4607 - val_loss: 2.3371 - val_sex_out_loss: 0.6499 - val_age_out_loss: 1.6872 - v
al_sex_out_acc: 0.6066 - val_age_out_acc: 0.3711
Epoch 25/25
16128/16128 [==============================] - 20s 1ms/step - loss: 2.2327 - sex_out_loss: 0.6383 - age_out_loss: 1.5943
- sex_out_acc: 0.6514 - age_out_acc: 0.4593 - val_loss: 2.4438 - val_sex_out_loss: 0.6779 - val_age_out_loss: 1.7659 - v
al_sex_out_acc: 0.5893 - val_age_out_acc: 0.3359
```

The current validation gender accuracy is **59%** and validation age group accuracy is **34%** which is not bad for a start.

2.3 TODOS

2.3.1 CREATE A GUI



I will create a GUI for the project to demonstrate its features using **PyQT** which is a library I have previously used and experienced on. The original frame will be on top-left, the frame with bboxes and class values will be on bottom-left, selected advertisement and running statistics will be on the right.

2.3.2 TUNE THE MODEL

I will try to change the losses, layers and architecture of my model to get better and relational accuracies.

2.3.3 FIND SAMPLE VIDEOS

I will find live-cam footages that include crowd in action. I will test my model on them to simulate real-life situations. Also I will find or create some sample advertise videos.