

# Final project appendix

Rayna liu, Zheyue Wang

April 2021

## 1 Appendix

```
1 # I. =====Load Libraries
   =====
2 import os
3 import cv2
4 import keras
5 import random
6 import numpy as np
7 import pandas as pd
8 import seaborn as sn
9 import sklearn.metrics
10 import tensorflow as tf
11 import matplotlib.pyplot as plt
12 from keras.models import load_model
13 from keras.models import Sequential
14 from keras.optimizers import Adam, SGD
15 from keras.utils import to_categorical
16 from sklearn.model_selection import train_test_split
17 from tensorflow.keras.preprocessing.image import ImageDataGenerator
18 from keras.callbacks import ModelCheckpoint, EarlyStopping,
   ReduceLROnPlateau
19 from keras.layers import Dense, Dropout, BatchNormalization, Conv2D,
   Flatten, MaxPooling2D, AveragePooling2D, InputLayer
20 from datetime import datetime
21
22 # The time of program's execution
23 start_time = datetime.now()           # Duration: 1:30:21.060016
24
25
26 # II. =====Load Dataset
   =====
```

```

27 data = pd.read_csv('age_gender.csv')
28 data.info()      # Check the type of data and NaN
29 data.head()      # Show the first 5 row of dataset
30
31
32 # III. =====Data Preprocessing (for Exploratory Data
    Analysis)=====
33 # Convert the type of pixels into np.array
34 data['pixels'] = data['pixels'].apply(lambda x: np.array(x.split(),
    dtype='float32'))
35 print('The pixels of each image is', len(data['pixels'][0]))
    # The length of pixels of each image: 2304 (48x48)
36 print('The image width is', int(np.sqrt(len(data['pixels'][0]))))
    # 48
37 print('The image height is', int(np.sqrt(len(data['pixels'][0]))))
    # 48
38
39
40 # IV. =====Description of dataset
    =====
41 data['Number of Image'] = 1
42
43 # Age
44 # Age 1-116, missing some age
45 Age_group = data.groupby(by='age')['Number of Image'].sum().
    reset_index(name='Number of Image')
46 # Plot the distribution of Age Group
47 plt.figure()
48 plt.bar(Age_group['age'], Age_group['Number of Image'])
49 plt.title('The Distribution of Age Group')
50 plt.xlabel('Age')
51 plt.ylabel('Number of Image')
52 plt.show()
53
54 # Ethnicity
55 # 0: White, 1:Black, 2:Asian, 3:Indian, 4:Others(like Hispanic,
    Latino, Middle Eastern)
56 Ethnicity_group = data.groupby(by='ethnicity')['Number of Image'].
    sum().reset_index(name='Number of Image')
57 # Plot the distribution of Ethnicity Group
58 label_ethnicity = ['White', 'Black', 'Asian', 'Indian', 'Others']
59 plt.figure()
60 plt.bar(Ethnicity_group['ethnicity'], Ethnicity_group['Number of
    Image'])
61 plt.title('The Distribution of Ethnicity Group')
62 plt.xlabel('Ethnicity')

```

```

63 plt.ylabel('Number of Image')
64 plt.xticks(Ethnicity_group['ethnicity'], label_ethnicity)
65 plt.show()
66
67 # Gender
68 # 0: Male, 1:Female
69 Gender_group = data.groupby(by='gender')['Number of Image'].sum().
    reset_index(name='Number of Image')
70 # Plot the distribution of Gender Group
71 label_gender = ['Male', 'Female']
72 plt.figure()
73 plt.bar(Gender_group['gender'], Gender_group['Number of Image'])
74 plt.title('The Distribution of Gender Group')
75 plt.xlabel('Gender')
76 plt.ylabel('Number of Image')
77 plt.xticks(Gender_group['gender'], label_gender)
78 plt.show()
79
80 # V. ----Show some sample images randomly-----
81 def sample_images(data):
82     fig, axs = plt.subplots(4,4,figsize=(16,16))
83     df = data.sample(n=16).reset_index(drop=True)
84     axs = axs.ravel()
85     j = 0
86     for i in range(len(df)):
87         if j < 16:
88             pixels = df['pixels'][i].reshape(48,48)
89             axs[j].imshow(pixels, cmap='gray')
90             axs[j].get_xaxis().set_ticks([])
91             axs[j].get_yaxis().set_ticks([])
92             axs[j].set_xlabel(f"Age: {df['age'].iloc[i]}, Ethnicity: {df['
ethnicity'].iloc[i]}, Gender: {'F' if df['gender'].iloc[i]==1
else 'M'}", fontsize=14)
93             j += 1
94         else:
95             break
96     fig.suptitle('Sample Images', fontsize=40)
97     plt.show()
98
99 ## Plot 16 images with label of age, ethnicity, and gender randomly
100 sample_images(data)
101
102
103 # VI. ===== Two Models
    =====
104 # Setting up

```

```

105 SEED = 42
106 os.environ['PYTHONHASHSEED'] = str(SEED)
107 random.seed(SEED)
108 np.random.seed(SEED)
109 tf.random.set_seed(SEED)
110
111 # ----- 1. Age Predict Model
112 # -----
113 # Hyper Parameters
114 LR_age = 0.001
115 N_EPOCHS_age = 200
116 BATCH_SIZE_age = 64
117 DROPOUT_age = 0.5
118
119 # Data Preparation
120 Image = np.array(data['pixels'].tolist())
121 Image = Image.reshape(Image.shape[0],48,48,1) # Reshapes to (
122         n_examples, n_channels, height_pixels, width_pixels)
123 Age_target = np.array(data['age']).reshape(-1,1)
124 Image = Image/255 # Normalizing
125         pixels data
126 print('Input Image shape:',Image.shape) # (23705, 48, 48,
127         1)
128 print('Age Target shape:',Age_target.shape) # (23705, 1)
129 print()
130
131 # Split Data into training set (70%), validation set (15%), and
132         testing set (15%)
133 age_x_train, age_x_test, age_y_train, age_y_test = train_test_split(
134         Image, Age_target, test_size=0.3, random_state=SEED, shuffle=True
135         ) # stratify=Age_target (ValueError: The least populated class
136         in y has only 1 member, which is too few. The minimum number of
137         groups for any class cannot be less than 2.)
138 age_x_val, age_x_test, age_y_val, age_y_test = train_test_split(
139         age_x_test, age_y_test, test_size=0.5, random_state=SEED, shuffle
140         =True)
141 print('Age train samples:',age_x_train.shape[0]) # 16593
142 print('Age validation samples:',age_x_val.shape[0]) # 3556
143 print()
144
145 # Image Augmentation (It didn't do well in modeling)
146 ## A technique to increase the diversity of your training set by
147         applying random (but realistic) transformations such as image
148         rotation and normalizing
149 #train_data_gen = ImageDataGenerator(rotation_range=40,
150         width_shift_range=1,brightness_range=[0.8,1.2],zoom_range

```

```

137     =[0.8,1.2], horizontal_flip=True, rescale=1/255)
138 #test_data_gen = ImageDataGenerator(rescale=1/255)
139 #age_train = train_data_gen.flow(age_x_train, age_y_train, seed=SEED
    , shuffle=False, batch_size=BATCH_SIZE_age)
140 #age_test = test_data_gen.flow(age_x_test, age_y_test, seed=SEED,
    shuffle=False, batch_size=BATCH_SIZE_age)
141
142 # Training Preparation
143 model_age = tf.keras.Sequential([
144     InputLayer(input_shape=(48,48,1)),
145     # output feature map size = ((
    image_size+2*pad)-(kernel_size-1)) / (max*stride)
146     Conv2D(64, (3, 3), strides=(1,1),
    padding='valid', activation="relu"),    # output(n_examples, 64,
    48, 48)
147     BatchNormalization(),
    MaxPooling2D((1, 1)),
    # output(n_examples, 64,
    48, 48)
148
149     Conv2D(128, (2, 2), activation="
    relu"),    # output(n_examples,
    128, 46, 46)
150     BatchNormalization(),
151     MaxPooling2D((2, 2)),
    # output(n_examples, 128,
    23, 23)
152
153     Conv2D(256, (3, 3), activation="
    relu"),    # output(n_examples,
    256, 21, 21)
154     BatchNormalization(),
155     MaxPooling2D((2, 2)),
    # output(n_examples, 256,
    10, 10)
156
157     Conv2D(512, (3, 3), activation="
    relu"),    # output(n_examples,
    512, 8, 8)
158     BatchNormalization(),
159     MaxPooling2D((2, 2)),
    # output(n_examples, 512,
    4, 4)
160
161     Conv2D(512, (3, 3),strides=(1, 1),
    padding='same', activation="relu"),    # output(n_examples, 512,

```

```

162         4, 4)
                                Conv2D(512, (3, 3), strides=(1, 1),
padding='same', activation="relu"),    # output(n_examples, 512,
163         4, 4)
                                Conv2D(512, (3, 3),strides=(1, 1),
padding='same', activation="relu"),    # output(n_examples, 512,
164         4, 4)
                                Conv2D(512, (3, 3),strides=(1, 1),
padding='same', activation="relu"),    # output(n_examples, 512,
165         4, 4)
                                Conv2D(512, (3, 3),strides=(1, 1),
padding='same', activation="relu"),    # output(n_examples, 512,
166         4, 4)
                                Conv2D(512, (3, 3),strides=(1, 1),
padding='same', activation="relu"),    # output(n_examples, 512,
167         4, 4)
                                Conv2D(512, (3, 3),strides=(1, 1),
padding='same', activation="relu"),    # output(n_examples, 512,
168         4, 4)
                                MaxPooling2D((2, 2)),
                                # output(n_examples, 512,
169         2, 2)
170                                Flatten(),
171                                Dense(512, activation="relu"),
172                                Dense(4096, activation="relu"),
173                                Dropout(DROPOUT_age),
174                                Dense(5000, activation="relu"),
175                                Dropout(DROPOUT_age),
176                                BatchNormalization(),
177                                Dense(1, activation="relu")
178                                ])
179 # Age Model Summary
180 model_age.summary()
181
182 # Compiling the model
183 model_age.compile(optimizer=Adam(lr=LR_age), loss="
mean_squared_error", metrics=["mae"])
184
185 # Setting callbacks
186 callbacks_age = [ModelCheckpoint("age_model_best.hdf5", monitor="
val_loss", save_best_only=True),
187                 EarlyStopping(monitor='val_loss', mode='min',
verbose=1, patience=25, restore_best_weights=True),
188                 ReduceLROnPlateau(factor=0.105, patience=15)]
189

```

```

190 # Training Loop
191 history_age = model_age.fit(age_x_train, age_y_train, batch_size=
    BATCH_SIZE_age, epochs=N_EPOCHS_age, validation_data=(age_x_val,
    age_y_val), callbacks=callbacks_age)
192 print()
193
194 # Evaluate Training History
195 plt.figure()
196 plt.plot(history_age.history['mae'])
197 plt.plot(history_age.history['val_mae'])
198 plt.title('Age model accuracy')
199 plt.ylabel('Mean Absolute Error')
200 plt.xlabel('Epoch')
201 plt.legend(['train mae', 'validation mae'], loc='upper right')
202 plt.show()
203
204 # Test MAE in age model
205 print("Test MAE of age:", model_age.evaluate(age_x_test, age_y_test)
    [1])
206 print()
207
208 # ----- 2. Ethnicity & Gender Predict
    Model-----
209 # Hyper Parameters
210 LR_eg = 0.0001
211 N_EPOCHS_eg = 60
212 BATCH_SIZE_eg = 512
213 DROPOUT_eg = 0.3
214 num_classes_eg = 7
215
216 # Data Preparation
217 Ethnicity_target = np.array(data['ethnicity']).reshape(-1,1)
218 Gender_target = np.array(data['gender']).reshape(-1,1)
219 Ethnicity_class = keras.utils.to_categorical(Ethnicity_target,5)
220 Gender_class = keras.utils.to_categorical(Gender_target,2)
221 # Concatenate Ethnicity and Gender target in one-hot-encoded (eg.
    White:0,Female:1 is [1,0,0,0,0,0,1])
222 Ethnicity_Gender_target = np.concatenate((Ethnicity_class,
    Gender_class),axis=1) # one-hot-encoded
223 print('Input Image shape:',Image.shape)
    # (23705, 48, 48, 1)
224 print('Ethnicity & Gender Target shape:',Ethnicity_Gender_target.
    shape) # (23705, 7)
225
226 # Split Data into training set (70%) and validation set (15%)
    testing set (15%)

```

```

227 eg_x_train, eg_x_test, eg_y_train, eg_y_test = train_test_split(
    Image, Ethnicity_Gender_target, test_size=0.3, random_state=SEED,
    stratify=Ethnicity_Gender_target)
228 eg_x_val, eg_x_test, eg_y_val, eg_y_test = train_test_split(
    eg_x_test, eg_y_test, test_size=0.5, random_state=SEED, stratify=
    eg_y_test)
229 print('Ethnicity & Gender train sample:',eg_x_train.shape[0])
    # 16593
230 print('Ethnicity & Gender validation sample:',eg_x_val.shape[0])
    # 3556
231
232 # Image Augmentation (It didn't do well in modeling)
233 ## A technique to increase the diversity of your training set by
    applying random (but realistic) transformations such as image
    rotation and normalizing
234 #train_data_gen = ImageDataGenerator(rotation_range=40,
    width_shift_range=0.2,brightness_range=[0.8,1.2],zoom_range
    =[0.8,1.2],rescale=1/255)
235 #test_data_gen = ImageDataGenerator(rescale=1/255)
236 #eg_train = train_data_gen.flow(eg_x_train, eg_y_train,batch_size=
    BATCH_SIZE_eg)
237 #eg_test = test_data_gen.flow(eg_x_test, eg_y_test,batch_size=
    BATCH_SIZE_eg)
238
239 # Training Preparation
240 model_eg = tf.keras.Sequential([
241     InputLayer(input_shape=(48,48,1)),
242     Conv2D(32,(3,3),activation="relu"),
    # output(n_examples, 32, 46, 46)
243     BatchNormalization(),
244     MaxPooling2D((2,2)),
    # output(n_examples, 32, 23, 23)
245
246     Conv2D(64,(3,3), activation="relu"),
    # output(n_examples, 64, 21, 21)
247     BatchNormalization(),
248     AveragePooling2D((2,2)),
    # output(n_examples, 64, 10, 10)
249
250     Conv2D(128,(3,3), activation="relu")
    ,
    # output(n_examples, 128, 8, 8)
251     BatchNormalization(),
252
253     Conv2D(200,(3,3), activation="relu")
    ,
    # output(n_examples, 200, 6, 6)
254     BatchNormalization(),

```



```

255         Conv2D(400,(3,3),activation='relu'),
256     # output(n_examples, 400, 4, 4)
257         BatchNormalization(),
258
259         Conv2D(500,(3,3),activation="relu"),
260     # output(n_examples, 500, 2, 2)
261
262         Flatten(),
263         Dense(500, activation="relu"),
264         Dropout(DROPOUT_eg),
265         BatchNormalization(),
266         Dense(4096,activation='relu'),
267         Dropout(DROPOUT_eg),
268         BatchNormalization(),
269         Dense(4096,activation="relu"),
270         Dropout(0.2),
271         BatchNormalization(),
272         Dense(4096,activation='relu'),
273         Dropout(0.2),
274         BatchNormalization(),
275         Dense(4096,activation='relu'),
276         Dropout(0.2),
277         BatchNormalization(),
278         Dense(4096,activation='relu'),
279         Dropout(0.2),
280         BatchNormalization(),
281         Dense(4096,activation='relu'),
282         Dropout(0.2),
283         BatchNormalization(),
284         Dense(4096,activation='relu'),
285         Dropout(0.2),
286         BatchNormalization(),
287         Dense(4096,activation='relu'),
288         Dropout(0.2),
289         BatchNormalization(),
290         Dense(4096,activation='relu'),
291         Dropout(0.2),
292         BatchNormalization(),
293         Dense(7, activation="sigmoid")
294     ])
295
296 # Ethnicity and Gender Model Summary
297 model_eg.summary()
298
299 # Compiling the model

```

```

299 model_eg.compile(optimizer='adam', loss=tf.keras.losses.
      BinaryCrossentropy(), metrics=["accuracy"])
300
301 # Setting callbacks
302 callbacks = [ModelCheckpoint("Eg_model.hdf5", monitor="val_accuracy"
      , save_best_only=True),
303               EarlyStopping(monitor='val_accuracy', mode='max',
      verbose=1, patience=30, restore_best_weights=True),
304               ReduceLROnPlateau(factor=0.105, patience=20)]
305
306 # Training Loop
307 history_eg = model_eg.fit(eg_x_train, eg_y_train, batch_size=
      BATCH_SIZE_eg, epochs=N_EPOCHS_eg, validation_data=(eg_x_test,
      eg_y_test), callbacks=callbacks)
308
309 # Evaluate Training History of Ethnicity & Gender Model
310 plt.figure()
311 plt.plot(history_eg.history['loss'])
312 plt.plot(history_eg.history['val_loss'])
313 plt.title('Ethnicity & Gender model accuracy')
314 plt.ylabel('Loss')
315 plt.xlabel('Epoch')
316 plt.legend(['train loss', 'validation loss'], loc='upper right')
317 plt.show()
318
319 # Test accuracy in ethnicity_gender model
320 print("Test accuracy on ethnicity and gender:", 100*model_eg.
      evaluate(eg_x_test, eg_y_test)[1], "%")
321
322
323 # VII. =====Age, Ethnicity, and Gender Prediction
      =====
324
325 # Predict age, ethnicity, and gender of all images
326 age_prediction = np.round(model_age.predict(Image)).astype(int)
327 ethnicity_gender_prediction = np.round(model_eg.predict(Image))
328 ethnicity_prediction = ethnicity_gender_prediction[:, :-2].argmax(
      axis=1)
329 gender_prediction = ethnicity_gender_prediction[:, -2:].argmax(axis
      =1)
330
331 # Create a DataFrame contain the prediction and predict error
332 data_pred = data.copy()
333 data_pred['age pred'] = age_prediction
334 data_pred['ethnicity pred'] = ethnicity_prediction
335 data_pred['gender pred'] = gender_prediction

```

```

336 data_pred['age error'] = data_pred['age']-data_pred['age pred']
337 data_pred['ethnicity error'] = data_pred['ethnicity']-data_pred['
    ethnicity pred']
338 data_pred['gender error'] = data_pred['gender']-data_pred['gender
    pred']
339
340 ## Plot some images with real label and predict label randomly
341 def predict_images(data):
342     fig, axs = plt.subplots(4,4,figsize=(16,16))
343     df = data_pred.sample(n=16).reset_index(drop=True)
344     axs = axs.ravel()
345     j = 0
346     for i in range(len(df)):
347         if j < 16:
348             pixels = df['pixels'][i].reshape(48,48)
349             axs[j].imshow(pixels, cmap='gray')
350             axs[j].get_xaxis().set_ticks([])
351             axs[j].get_yaxis().set_ticks([])
352             axs[j].set_xlabel('[Real]: Age:'+str(df['age'].iloc[i])+
353                             ' Ethnicity:'+str(df['ethnicity'].iloc[i])+
354                             ' Gender:'+str(df['gender'].iloc[i])+
355                             '\n[Pred]: Age:'+str(df['age pred'].iloc[i])
356                             +
357                             ' Ethnicity:'+str(df['ethnicity pred'].iloc[
358                                 i])+
359                             ' Gender:'+str(df['gender pred'].iloc[i]))
360             j += 1
361         else:
362             break
363     fig.suptitle('Sample Prediction Images',fontsize=40)
364     plt.show()
365
366 predict_images(data_pred)
367
368 # VII. =====Prediction Accuracy Analysis
369     =====
370
371 # -----1. Age
372     -----
373 Age_pred_group = data_pred[data_pred['age']==data_pred['age pred']]
374 Age_correct_group = Age_pred_group.groupby(by='age')['Number of
    Image'].sum().reset_index(name='Sum of Correct Image')
375 age_result = pd.merge(Age_group, Age_correct_group, on='age')
376 age_result['Accuracy'] = 100*age_result['Sum of Correct Image']/
    age_result['Number of Image']
377 conditions = [(age_result['age'] <= 10),

```

```

374         (age_result['age'] <= 20) & (age_result['age'] > 10),
375         (age_result['age'] <= 30) & (age_result['age'] > 20),
376         (age_result['age'] <= 40) & (age_result['age'] > 30),
377         (age_result['age'] <= 50) & (age_result['age'] > 40),
378         (age_result['age'] <= 60) & (age_result['age'] > 50),
379         (age_result['age'] <= 70) & (age_result['age'] > 60),
380         (age_result['age'] <= 80) & (age_result['age'] > 70),
381         (age_result['age'] <= 90) & (age_result['age'] > 80),
382         (age_result['age'] <= 100) & (age_result['age'] > 90),
383         (age_result['age'] > 100)]
384 values = ['0-10', '11-20', '21-30', '31-40', '41-50', '51-60', '61-70',
385           ', '71-80', '81-90', '91-100', '100+']
386 age_result['Age Groups'] = np.select(conditions, values)
387 Age_Group_avg = age_result.groupby(by='Age Groups')['Accuracy'].mean(
388     ).reset_index(name='Average of Accuracy')
389
390 # Plot the Distribution of Correctly Prediction of Whole Age Group
391 plt.figure()
392 plt.bar(Age_Group_avg['Age Groups'], Age_Group_avg['Average of
393     Accuracy'], color=['palevioletred']) # change x to
394     age_result['age'] could see the distribution of correctly
395     prediction of age
396 plt.title('The Distribution of Correctly Prediction of Whole Age
397     Group')
398 plt.xlabel('Age Groups')
399 plt.ylabel('Percentage of Correct Prediction Image (%)')
400 for a,b in zip(Age_Group_avg['Age Groups'], Age_Group_avg['Average of
401     Accuracy']):
402     plt.text(a, b+0.05, '%.3f' % b, ha='center', va= 'bottom',
403         fontsize=7)
404 plt.show()
405
406 # Histogram of prediction error of age
407 plt.figure()
408 plt.hist(data_pred['age error'], bins=40)
409 plt.title('The Histogram of Prediction Error of Age')
410 plt.xlabel('Error')
411 plt.ylabel('Number of Images')
412 plt.xlim(-20,20)
413 plt.show()
414
415 # ----- 2. Ethnicity
416 -----
417 Ethnicity_pred_group = data_pred[data_pred['ethnicity']==data_pred['
418     ethnicity pred']]

```

```

410 Ethnicity_correct_group = Ethnicity_pred_group.groupby(by='ethnicity
    ')[ 'Number of Image' ].sum().reset_index(name='Sum of Correct
    Image')
411 ethnicity_result = pd.merge(Ethnicity_group, Ethnicity_correct_group
    , on='ethnicity')
412 ethnicity_result['Accuracy'] = 100*ethnicity_result['Sum of Correct
    Image']/ethnicity_result['Number of Image']
413
414 # Plot the Distribution of Correctly Prediction of Ethnicity Group
415 label_ethnicity = ['White', 'Black', 'Asian', 'Indian', 'Others']
416 plt.figure()
417 plt.bar(ethnicity_result['ethnicity'], ethnicity_result['Accuracy'])
418 plt.title('The Distribution of Correctly Prediction of Ethnicity
    Group')
419 plt.xlabel('Ethnicity')
420 plt.ylabel('Percentage of Correct Prediction Image (%)')
421 plt.xticks(ethnicity_result['ethnicity'], label_ethnicity)
422 plt.show()
423
424 # Plot Confusion Matrix of Ethnicity
425 cm = sklearn.metrics.confusion_matrix(data_pred['ethnicity'],
    data_pred['ethnicity pred'])
426 df_cm = pd.DataFrame(cm, range(5), range(5))
427 plt.figure(figsize=(10,7))
428 sn.set(font_scale=1.4) # for label size
429 sn.heatmap(df_cm, annot=True, annot_kws={"size": 16}, cmap="Blues")
    # font size
430 plt.xlabel("Predicted Ethnicity")
431 plt.ylabel("True Ethnicity")
432 plt.title('Confusion Matrix of Ethnicity')
433 plt.show()
434
435
436 # -----3. Gender
    -----
437 Gender_pred_group = data_pred[data_pred['gender']==data_pred['gender
    pred']]
438 Gender_correct_group = Gender_pred_group.groupby(by='gender')['
    Number of Image'].sum().reset_index(name='Sum of Correct Image')
439 gender_result = pd.merge(Gender_group, Gender_correct_group, on='
    gender')
440 gender_result['Accuracy'] = 100*gender_result['Sum of Correct Image'
    ]/gender_result['Number of Image']
441
442 # Plot the Distribution of Correctly Prediction of Gender Group
443 label_gender = ['Male', 'Female']

```

```

444 plt.figure(figsize=(10,7))
445 plt.bar(gender_result['gender'],gender_result['Accuracy'])
446 plt.title('The Distribution of Correctly Prediction of Gender Group'
447 )
448 plt.xlabel('Gender')
449 plt.ylabel('Percentage of Correct Prediction Image (%)')
450 plt.xticks(gender_result['gender'],label_gender)
451 plt.show()
452 # Plot Confusion Matrix of Gender
453 cm = sklearn.metrics.confusion_matrix(data_pred['gender'],data_pred[
454     'gender pred'])
455 df_cm = pd.DataFrame(cm, range(2), range(2))
456 plt.figure(figsize=(10,7))
457 sn.set(font_scale=1.4) # for label size
458 sn.heatmap(df_cm, annot=True, annot_kws={"size": 16},cmap="Blues")
459 # font size
460 plt.xlabel("Predicted Gender")
461 plt.ylabel("True Gender")
462 plt.title('Confusion Matrix of Gender')
463 plt.show()
464 # VIII.===== Error Analysis
465 # Show Incorrectly Age Images
466 def Incorrectly_images_age(x_test,y_test):
467     fig, axs = plt.subplots(4,4,figsize=(16,16))
468     age_y_pred = np.round(model_age.predict(x_test))
469     axs = axs.ravel()
470     x_test = x_test * 255
471     j = 0
472     for i in range(len(x_test)):
473         if j < 16:
474             if age_y_pred[i] != y_test[i]:
475                 pixels = x_test[i].reshape((48,48))
476                 axs[j].imshow(pixels, cmap='gray')
477                 axs[j].get_xaxis().set_ticks([])
478                 axs[j].get_yaxis().set_ticks([])
479                 axs[j].set_xlabel('Real Age:'+str(y_test[i])+
480                     ' Pred Age:'+str(age_y_pred[i]),
481                     fontsize=14)
482                 j += 1
483             else:
484                 break
485     fig.suptitle('Incorrectly Age Images',fontsize=40)

```

```

485     plt.show()
486
487 Incorrectly_images_age(age_x_test, age_y_test)
488
489 # Show Incorrectly Ethnicity and Gender Images
490 eg_y_pred = model_eg.predict(eg_x_test)
491 eg_y_pred = np.round(eg_y_pred)
492 ethnicity_pred = eg_y_pred[:, :-2].argmax(axis=1)
493 gender_pred = eg_y_pred[:, -2:].argmax(axis=1)
494
495 ethnicity_y_test = eg_y_test[:, :-2].argmax(axis=1)
496 gender_y_test = eg_y_test[:, -2:].argmax(axis=1)
497
498 def Incorrectly_images_eg(x_test, y_test):
499     fig, axs = plt.subplots(4, 4, figsize=(16, 16))
500     axs = axs.ravel()
501     x_test = x_test * 255
502     j = 0
503     for i in range(len(x_test)):
504         if j < 16:
505             if ethnicity_pred[i] != ethnicity_y_test[i] and
gender_pred[i] != gender_y_test[i]:
506                 pixels = x_test[i].reshape((48, 48))
507                 axs[j].imshow(pixels, cmap='gray')
508                 axs[j].get_xaxis().set_ticks([])
509                 axs[j].get_yaxis().set_ticks([])
510                 axs[j].set_xlabel('[Real] Ethnicity:' + str(
ethnicity_y_test[i]) +
511                                     ' Gender:' + str(gender_y_test[i]) +
512                                     '\n [Pred] Ethnicity:' + str(
ethnicity_pred[i]) +
513                                     ' Gender:' + str(gender_pred[i]),
                    fontsize=14)
514                 j += 1
515             else:
516                 break
517     fig.suptitle('Incorrectly Images', fontsize=40)
518     plt.show()
519
520 Incorrectly_images_eg(eg_x_test, eg_y_test)
521
522 # IX.===== Application
=====
523 # Predict the age, ethnicity, and gender of the image which randomly
choose from Google
524 # Load the Test_Images Folder

```

```

525 direction = os.getcwd() + "/Test_Images/"           # get the image
    path
526 image_path = []
527 for im in [f for f in os.listdir(direction)]:
528     image_path.append(direction+im)
529
530 # Reshape the Images
531 RESIZE_TO = 48
532 x = []
533 pixels = []
534 for png in image_path:
535     pixel = cv2.imread(png)
536     pixels.append(pixel)
537     image = cv2.resize(pixel, (RESIZE_TO, RESIZE_TO))
538     image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
539     x.append(image)
540 x = np.array(x)
541 x = x.reshape(len(x), RESIZE_TO, RESIZE_TO, 1)
542 x = x / 255
543
544 # Age Prediction
545 age_pred_apply = np.round(model_age.predict(x))
546
547 # Ethnicity & Gender Prediction
548 ethnicity_gender_pred_apply = np.round(model_eg.predict(x))
549 ethnicity_pred_apply = ethnicity_gender_pred_apply[:, :-2].argmax(
    axis=1)
550 gender_pred_apply = ethnicity_gender_pred_apply[:, -2:].argmax(axis
    =1)
551
552 ethnicity_pred_label = []
553 for i in range(len(ethnicity_pred_apply)):
554     if ethnicity_prediction[i] == 0:
555         ethnicity_pred_label.append('White')
556     elif ethnicity_prediction[i] == 1:
557         ethnicity_pred_label.append('Black')
558     elif ethnicity_prediction[i] == 2:
559         ethnicity_pred_label.append('Asian')
560     elif ethnicity_prediction[i] == 3:
561         ethnicity_pred_label.append('Indian')
562     else:
563         ethnicity_pred_label.append('Other')
564
565 gender_pred_label = []
566 for i in range(len(gender_pred_apply)):
567     if gender_prediction[i] == 0:

```



```

568         gender_pred_label.append('M')
569     else:
570         gender_pred_label.append('F')
571
572 # Plot the Prediction
573 fig, axs = plt.subplots(3,2,figsize=(16,16))
574 axs = axs.ravel()
575 for i in range(len(image_path)):
576     image = pixels[i]
577     axs[i].imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
578     axs[i].get_xaxis().set_ticks([])
579     axs[i].get_yaxis().set_ticks([])
580     axs[i].set_xlabel('[Pred]: Age:'+str(age_pred_apply[i])+
581                       ' Ethnicity:'+str([ethnicity_pred_label[i]])+
582                       ' Gender:'+str([gender_pred_label[i]]), fontsize
583                               =18)
584 fig.suptitle('Prediction Images from Google Image',fontsize=40)
585 plt.show()
586
587 # End time
588 end_time = datetime.now()
589 print('Duration: {}'.format(end_time - start_time))

```