**Data description**

The dataset was obtained from [*ChaLearn Looking at People*](http://chalearnlap.cvc.uab.es/dataset/26/description/)*.* It was prepared for the project and placed in the /datasets/faces/ folder, there you can find

* The final\_files folder with 7.6k photos
* The labels.csv file with labels, with two columns: file\_name and real\_age

As the number of image files is rather high, it is advisable to avoid reading them all at once, which would be resource consuming, and to read them sequentially with the [*ImageDataGenerator*](https://keras.io/preprocessing/image/) generator. This method was explained in Chapter 3. Convolutional neural networks, Lesson 7. Data Generators.

**Your task**

Perform the exploratory data analysis:

* Look at the dataset size.
* Explore the age distribution in the dataset.
* Print 10-15 photos for different ages on the screen to get an overall impression of the dataset.

Paths to the files for analysis:

* '/datasets/faces/labels.csv'
* '/datasets/faces/final\_files/'

Provide findings on how the specifics of the dataset you discover may affect how the model is trained.

Save the notebook under a meaningful name.

# Model Training

Now that you’ve completed the exploratory data analysis, you’re ready to build and train your model. To do this, you can create functions like we did earlier in the sprint to load the data, build the model, and train it. For example:

* load\_train(path)
  + it loads the train dataset
* load\_test(path)
  + it loads the test dataset
* create\_model(input\_shape)
  + it defines a model
* train\_model(model, train\_data, test\_data, batch\_size, epochs, steps\_per\_epoch, validation\_steps)

To train the model in any reasonable amount of time, you would need to use a GPU. However, we no longer have GPUs available on the platform at this time. So instead of training the model yourself, we will provide with the results of a model we trained ourselves over 20 epochs.

You can also download the data for local usage by following [this link](https://practicum-content.s3.us-west-1.amazonaws.com/data-scientist/datasets/faces.zip).

Here is the model we built:

def create\_model(input\_shape):

"""

It defines model

"""

backbone = ResNet50(weights='imagenet',

input\_shape=input\_shape,

include\_top=False)

model = Sequential()

model.add(backbone)

model.add(GlobalAveragePooling2D())

model.add(Dense(1, activation='relu'))

optimizer = Adam(learning\_rate=0.0005)

model.compile(optimizer=optimizer, loss='mse', metrics=['mae'])

return model

Here is how we trained it:

def train\_model(model, train\_data, test\_data, batch\_size=None, epochs=20,

steps\_per\_epoch=None, validation\_steps=None):

"""

Trains the model given the parameters

"""

if steps\_per\_epoch is None:

steps\_per\_epoch = len(train\_data)

if validation\_steps is None:

validation\_steps = len(test\_data)

model.fit(train\_data,

validation\_data=test\_data,

batch\_size=batch\_size, epochs=epochs,

steps\_per\_epoch=steps\_per\_epoch,

validation\_steps=validation\_steps,

verbose=2)

return model

And here are the results of its training:

Epoch 1/20

356/356 - 35s - loss: 95.3532 - mae: 7.4339 - val\_loss: 124.3362 - val\_mae: 8.4921

Epoch 2/20

356/356 - 35s - loss: 76.8372 - mae: 6.6707 - val\_loss: 127.6357 - val\_mae: 8.6035

Epoch 3/20

356/356 - 35s - loss: 69.9428 - mae: 6.3992 - val\_loss: 91.1531 - val\_mae: 7.4454

Epoch 4/20

356/356 - 35s - loss: 64.4249 - mae: 6.1407 - val\_loss: 124.0287 - val\_mae: 8.3481

Epoch 5/20

356/356 - 35s - loss: 52.8486 - mae: 5.5913 - val\_loss: 109.1004 - val\_mae: 8.2192

Epoch 6/20

356/356 - 35s - loss: 46.3094 - mae: 5.2223 - val\_loss: 85.1038 - val\_mae: 7.0332

Epoch 7/20

356/356 - 35s - loss: 38.2617 - mae: 4.7951 - val\_loss: 92.0900 - val\_mae: 7.3359

Epoch 8/20

356/356 - 35s - loss: 37.4804 - mae: 4.7402 - val\_loss: 80.0016 - val\_mae: 6.7239

Epoch 9/20

356/356 - 35s - loss: 33.5237 - mae: 4.4271 - val\_loss: 83.2579 - val\_mae: 6.8529

Epoch 10/20

356/356 - 35s - loss: 28.5170 - mae: 4.1411 - val\_loss: 83.5056 - val\_mae: 6.9629

Epoch 11/20

356/356 - 35s - loss: 27.0142 - mae: 3.9700 - val\_loss: 92.1290 - val\_mae: 7.1866

Epoch 12/20

356/356 - 35s - loss: 27.4564 - mae: 4.0428 - val\_loss: 185.6307 - val\_mae: 11.4591

Epoch 13/20

356/356 - 35s - loss: 23.7961 - mae: 3.7407 - val\_loss: 92.3429 - val\_mae: 7.2467

Epoch 14/20

356/356 - 35s - loss: 24.6167 - mae: 3.8116 - val\_loss: 92.4542 - val\_mae: 7.1401

Epoch 15/20

356/356 - 35s - loss: 22.2604 - mae: 3.6746 - val\_loss: 82.5822 - val\_mae: 6.7841

Epoch 16/20

356/356 - 35s - loss: 20.1899 - mae: 3.4430 - val\_loss: 86.3830 - val\_mae: 6.8304

Epoch 17/20

356/356 - 35s - loss: 17.3425 - mae: 3.2205 - val\_loss: 78.4369 - val\_mae: 6.6419

Epoch 18/20

356/356 - 35s - loss: 16.5249 - mae: 3.1295 - val\_loss: 81.7731 - val\_mae: 6.7226

Epoch 19/20

356/356 - 35s - loss: 16.6140 - mae: 3.1421 - val\_loss: 80.9727 - val\_mae: 6.9908

Epoch 20/20

356/356 - 35s - loss: 17.0187 - mae: 3.1785 - val\_loss: 93.4115 - val\_mae: 7.6512

Copy these results into your Jupyter notebook in the next lesson and analyze what they say about the data and the model.

# Model Analysis

Add the output for the model training (as returned from the GPU platform) to the Jupyter notebook you've created at the EDA stage so as to put everything in one place. Then analyze the result of model training.

Can computer vision help the customer in this case?

What other practical tasks might the customer solve with the model? Feel free to share your ideas on this.