Springboard: Data Science Career Track
Capstone Project III: Project Report
By Lucien Meteumba
December, 2020

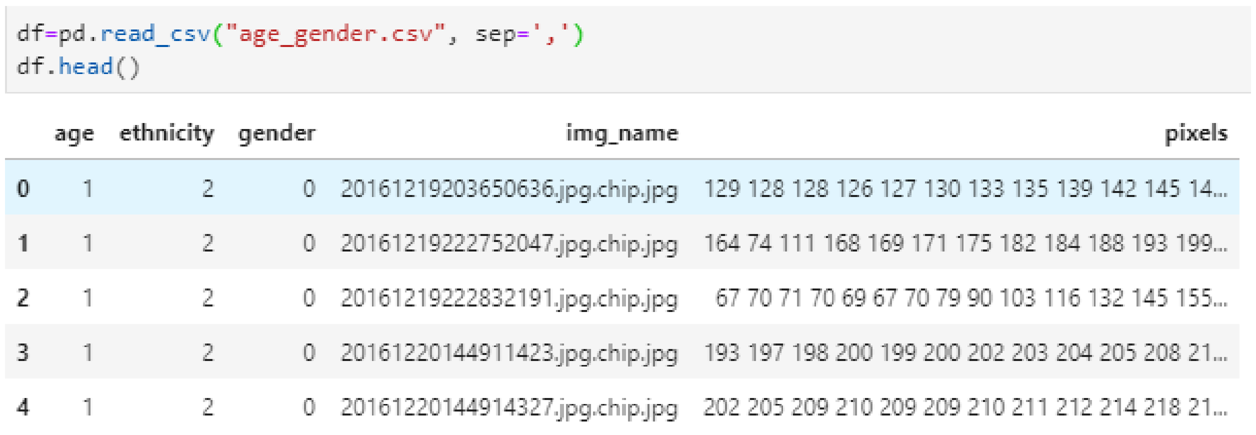
I- PROBLEM IDENTIFICATION OVERVIEW

The goal of this project is to use someone's picture (who wants to get into a soccer academy) to detect his real age. With the use of historical data, we can use deep learning to help us give the right age of someone. The reason why we need to detect the real age is because there has been a problem where people cut their age in order to have acceptance into a soccer academy or school. Another problem we will try to solve in this project will be to determine the age of old people whose birthdate is not known due to the fact that back in the days most people did not have the luxure to establish a birth certificate for their children. By scanning their image we will be able to tell how old they are.
According to wikidepia, Age fraud is age fabrication or the use of false documentation to gain an advantage over opponents. Since in most African countries, records are difficult to verify, African wanting to join a European soccer academy who do not have the age requirement can easily change their age to meet the requirement. This is a serious problem mostly in academy soccer in Europe because most Africans that are being recruited do not meet the age requirement. With the help of Machine Learning and Deep Learning we can now use their photo picture to identify how old they are.That will help Academy Sport to recruit the right people.
Due to the lack of technology and the lack of money in the past (from 1965 downward) in Africa, people were not able to establish a birth certificate for their children. That is why we see most of the old age people in Africa not knowing their age. The model we will be developing will help identify their age.
Our criteria for success will be to build, and evaluate the performance of a model that will be able to predict someone's age based on their image.
We will analyze the data taken from “age_gender.csv”[1] from Kaggle’s Website.
The main constraints in this project might be missing values and the fact that our dataset contains less african ethnicity.
The intended stakeholders are the European soccer clubs, African society.

II-DATA

[1] -https://www.kaggle.com/nipunarora8/age-gender-and-ethnicity-face-data-csv

The dataset we are using to support this project is “age_gender.csv” from Kaggle’s website. This dataset includes a CSV of facial images that are labeled on the basis of age, gender, and ethnicity.The dataset includes 27305 rows and 5 columns.
‘age’ is an integer from 0 to 116, indicating the age.
‘gender’ is either 0 (male) or 1 (female).
‘race’ is an integer from 0 to 4, denoting White, Black, Asian, Indian, and Others (like Hispanic, Latino, Middle Eastern).



```
df=pd.read_csv("age_gender.csv", sep=',')
df.head()
```

	age	ethnicity	gender	img_name	pixels
0	1	2	0	20161219203650636.jpg.chip.jpg	129 128 128 126 127 130 133 135 139 142 145 14...
1	1	2	0	20161219222752047.jpg.chip.jpg	164 74 111 168 169 171 175 182 184 188 193 199...
2	1	2	0	20161219222832191.jpg.chip.jpg	67 70 71 70 69 67 70 79 90 103 116 132 145 155...
3	1	2	0	20161220144911423.jpg.chip.jpg	193 197 198 200 199 200 202 203 204 205 208 21...
4	1	2	0	20161220144914327.jpg.chip.jpg	202 205 209 210 209 209 210 211 212 214 218 21...

III- DATA WRANGLING AND CLEANING

This dataset was a clean dataset. We did not find any outlier, any missing values, and no duplicates rows.

```
df.isnull().sum()
```

```
age          0
ethnicity    0
gender       0
img_name     0
pixels       0
dtype: int64
```

We can see that there is no missing value in the dataset.

Let us check if there are any duplicate row in our dataset

```
dupli=df.duplicated()
j=0
for i in dupli:
    if i==True:
        j=j+1
print('The number of duplicated rows are', j)
```

```
The number of duplicated rows are 0
```

We have zero duplicates rows, which shows us that our data is clean.

IV- EXPLORATORY DATA ANALYSIS

We started by reshaping our pixels columns.

```
#Let us convert pixels into numpy array
df['pixels']=df['pixels'].apply(lambda x: np.array(x.split(), dtype="float32").reshape(48, 48))
df['pixels']
```

```
0        [[129.0, 128.0, 128.0, 126.0, 127.0, 130.0, 13...
1        [[164.0, 74.0, 111.0, 168.0, 169.0, 171.0, 175...
2        [[67.0, 70.0, 71.0, 70.0, 69.0, 67.0, 70.0, 79...
3        [[193.0, 197.0, 198.0, 200.0, 199.0, 200.0, 20...
4        [[202.0, 205.0, 209.0, 210.0, 209.0, 209.0, 21...
                               ...
23700    [[127.0, 100.0, 94.0, 81.0, 77.0, 77.0, 74.0, ...
23701    [[23.0, 28.0, 32.0, 35.0, 42.0, 47.0, 68.0, 85...
23702    [[59.0, 50.0, 37.0, 40.0, 34.0, 19.0, 30.0, 10...
23703    [[45.0, 108.0, 120.0, 156.0, 206.0, 197.0, 140...
23704    [[156.0, 161.0, 160.0, 165.0, 170.0, 173.0, 16...
Name: pixels, Length: 23705, dtype: object
```

Then we normalized our pixels data.

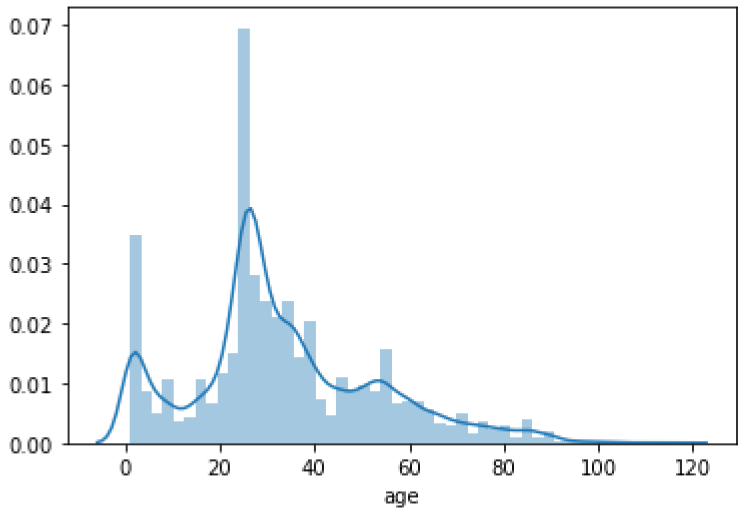
```
#We normalize the pixels data
df['pixels']=df['pixels'].apply(lambda x: x/255)
df['pixels'].head()
```

```
0    [[0.5058824, 0.5019608, 0.5019608, 0.49411765,...
1    [[0.6431373, 0.2901961, 0.43529412, 0.65882355...
2    [[0.2627451, 0.27450982, 0.2784314, 0.27450982...
3    [[0.75686276, 0.77254903, 0.7764706, 0.7843137...
4    [[0.7921569, 0.8039216, 0.81960785, 0.8235294,...
Name: pixels, dtype: object
```

After normalizing the pixels data, we decided to plot the age distribution.

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x17d12297948>
```

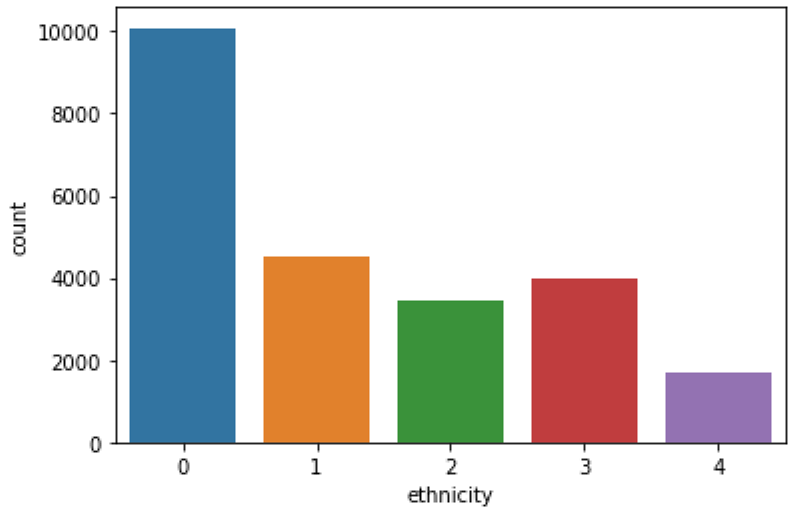


We can see that there are many people in our dataset that are between 0-2 years old and 24-26 years old. We can also observe that there are few people whose age is between 80-120.

This is what the ethnicity distribution gave us.

```
sns.countplot(df['ethnicity'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x17d1223e8c8>
```



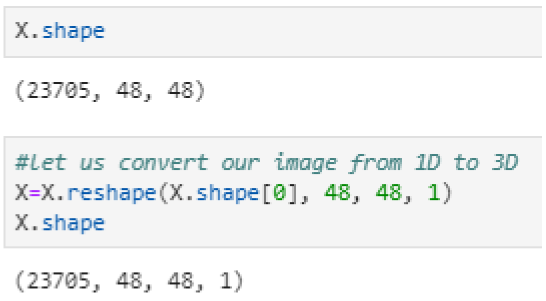
We could see that in this dataset there are more white than any other ethnicity. The black ethnicity comes in the second position. To be more accurate, 19.09% constitute the percentage of black people in this dataset.
We decided to visualize some of the images in the dataset. By displaying faces and genders we will be able to identify any anomaly in labeling the data.

```
# Plot some pictures
fig, axes = plt.subplots(1, 5, figsize=(20, 10))

for i in range(5):
    random_face = np.random.choice(len(df))

    age = df['age'][random_face]
    ethnicity = df['ethnicity'][random_face]
    gender = df['gender'][random_face]

    axes[i].set_title('Age: {0}, Ethnicity: {1}, Sex: {2}'.format(age, ethnicity, gender))
    axes[i].imshow(df['pixels'][random_face])
    axes[i].axis('off')
```

Age: 61, Ethnicity: 1, Sex: 0
Age: 45, Ethnicity: 3, Sex: 0
Age: 38, Ethnicity: 1, Sex: 0
Age: 28, Ethnicity: 1, Sex: 1
Age: 1, Ethnicity: 0, Sex: 1

V- PRE_PROCESSING AND TRAINING DATA DEVELOPMENT

In order to preprocess our data, we decided to reshape it.



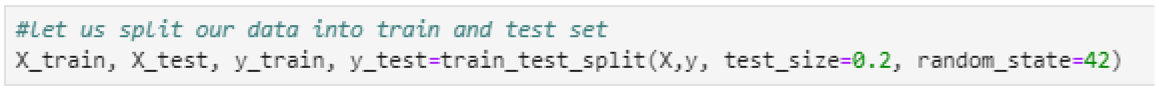
```
X.shape
```

```
(23705, 48, 48)
```

```
#Let us convert our image from 1D to 3D
X=X.reshape(X.shape[0], 48, 48, 1)
X.shape
```

```
(23705, 48, 48, 1)
```

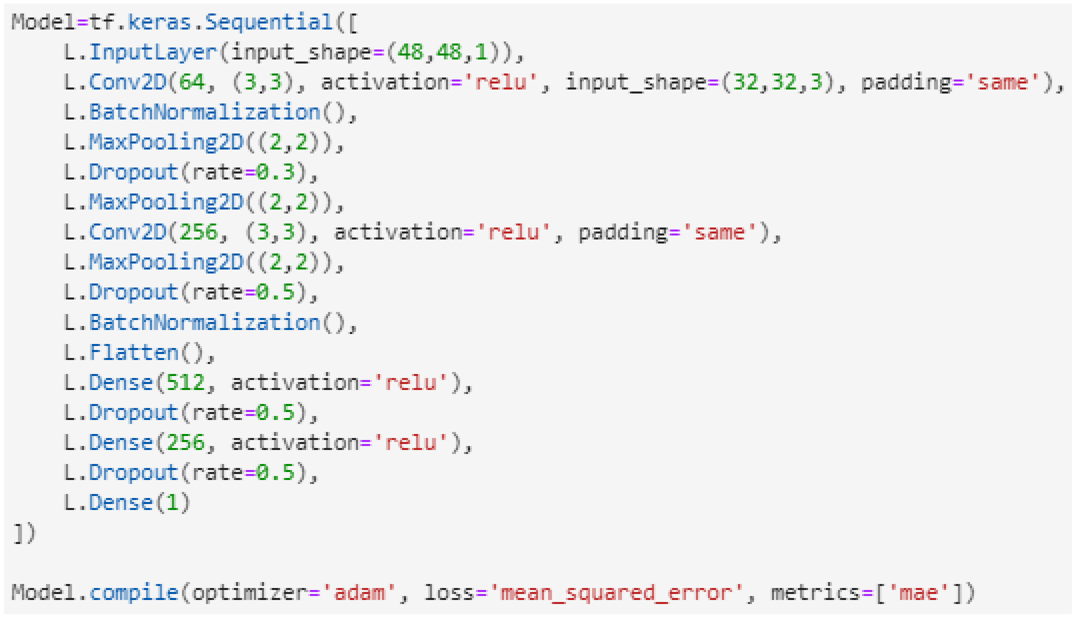
Then we split our data into train and test set



```
#Let us split our data into train and test set
X_train, X_test, y_train, y_test=train_test_split(X,y, test_size=0.2, random_state=42)
```

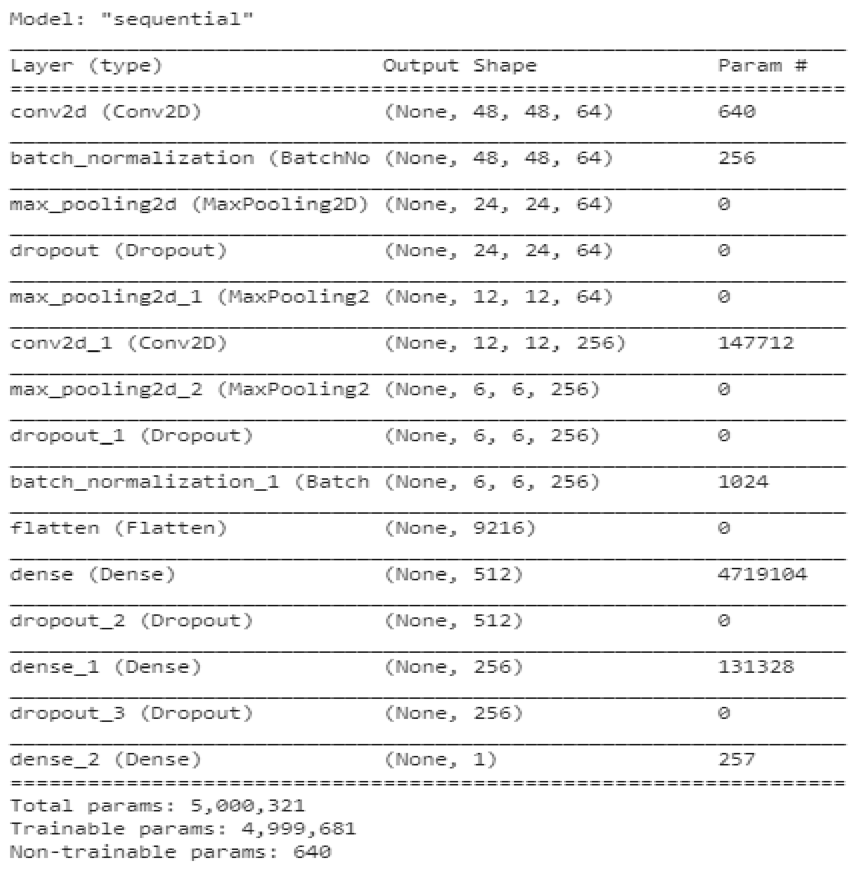
In Keras, we can just stack up layers by adding the desired layer one by one. To build our model we first of all used as input layer our image pixel with 48X48. Then the second layer was the convolutional layer with Conv2D.This layer is sometimes called feature extractor layer because features of the image are getting extracted within this layer. The third layer was to normalize the input layer by adjusting and scaling the activations using BatchNormalization(). The fourth layer down-sampled an input given by the third layer, reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. Here we used MaxPoolingg2D((2,2)) to do that. In the fifth layer we used Dropout(learning rate=0.3) to randomly select 30% of the neurons and set their weights to zero. Then we used MaxPoolingg2D((2,2)) in the sixth layer and in the seventh layer we used Con2D to learn a total of 256 filters and then we used Max Pooling to reduce the spatial dimensions of the output volume. In the eight layer we decided to randomly select 50% of the neurons to set their weights to zero, then we applied BatchNormalization() in the ninth layer. After that, we collapsed the input coming from the ninth layer into one dimension using Flatten(). The next layer was a Dense layer with 512 neurons where each neuron receives input from all the neurons in the previous layer, thus densely connected. In the eleventh, twelfth and thirteen layers we used respectively Dropout(learning rate=0.6), Dense(256, activation='relu') and Dropout(learning rate=0.4). Our output was Dense(1).

After building our model we compiled it with “adam” as the optimizer, “mean_squared_error” as the loss, and “mae” as the metrics.



```
Model=tf.keras.Sequential([
    L.InputLayer(input_shape=(48,48,1)),
    L.Conv2D(64, (3,3), activation='relu', input_shape=(32,32,3), padding='same'),
    L.BatchNormalization(),
    L.MaxPooling2D((2,2)),
    L.Dropout(rate=0.3),
    L.MaxPooling2D((2,2)),
    L.Conv2D(256, (3,3), activation='relu', padding='same'),
    L.MaxPooling2D((2,2)),
    L.Dropout(rate=0.5),
    L.BatchNormalization(),
    L.Flatten(),
    L.Dense(512, activation='relu'),
    L.Dropout(rate=0.5),
    L.Dense(256, activation='relu'),
    L.Dropout(rate=0.5),
    L.Dense(1)
])

Model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])
```

Our model summary is shown in the picture below



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 48, 48, 64)        640       
_________________________________________________________________
batch_normalization (BatchNo (None, 48, 48, 64)        256       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 24, 24, 64)        0         
_________________________________________________________________
dropout (Dropout)            (None, 24, 24, 64)        0         
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 12, 12, 256)       147712    
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 256)         0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 6, 6, 256)         0         
_________________________________________________________________
batch_normalization_1 (Batch (None, 6, 6, 256)         1024      
_________________________________________________________________
flatten (Flatten)            (None, 9216)              0         
_________________________________________________________________
dense (Dense)                (None, 512)               4719104   
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328    
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257       
=================================================================
Total params: 5,000,321
Trainable params: 4,999,681
Non-trainable params: 640
```

Now that our model was ready to be trained, we fitted our model by passing the input features X_train and y_train. We also passed the validation set equal to 10% of the training set and 20 epochs.

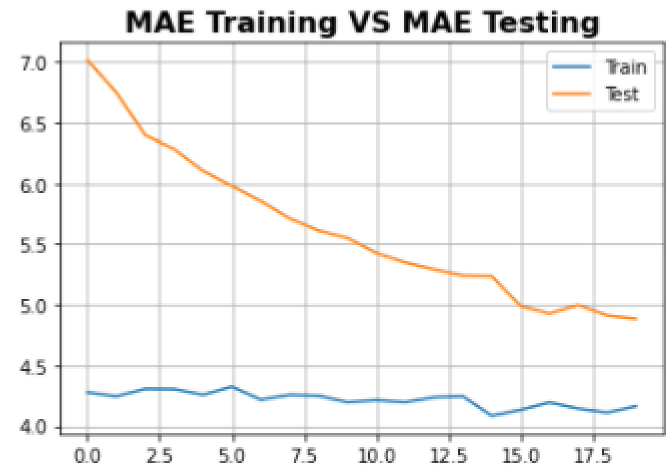
```
history=Model.fit(X_train, y_train, epochs=20, validation_split=0.1, batch_size=64, callbacks=[callback])
```

We decided to do the same operation with the testing set ie X_test and y_test

```
history1=Model.fit(X_test, y_test, epochs=20, validation_split=0.1, batch_size=64, callbacks=[callback])
```

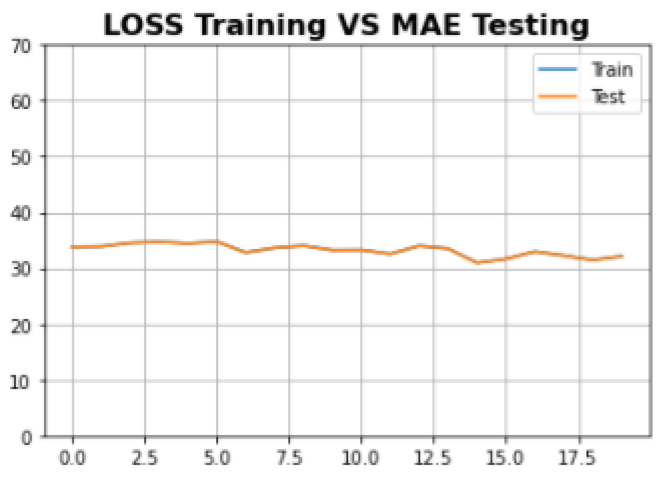
The goal of this last line of code is to plot the MAE of the training data VS MAE of the testing data. We are trying to see if the two lines are converging.

```
plt.plot(hist_train1, label="Train")
plt.plot(hist_test1, label="Test")
plt.legend(loc='upper right')
plt.grid(True)
plt.title("MAE Training VS MAE Testing", fontsize=16, fontweight='bold')
plt.show()
```



We could see that indeed they are converging towards each other.
We also plotted the LOSS of the training data VS testing data as we can see below.

```
plt.plot(hist_train2, label="Train")
plt.plot(hist_test2, label="Test")
plt.legend(loc='upper right')
plt.grid(True)
plt.title("LOSS Training VS MAE Testing", fontsize=16, fontweight='bold')
plt.gca().set_ylim(0,70)
plt.show()
```



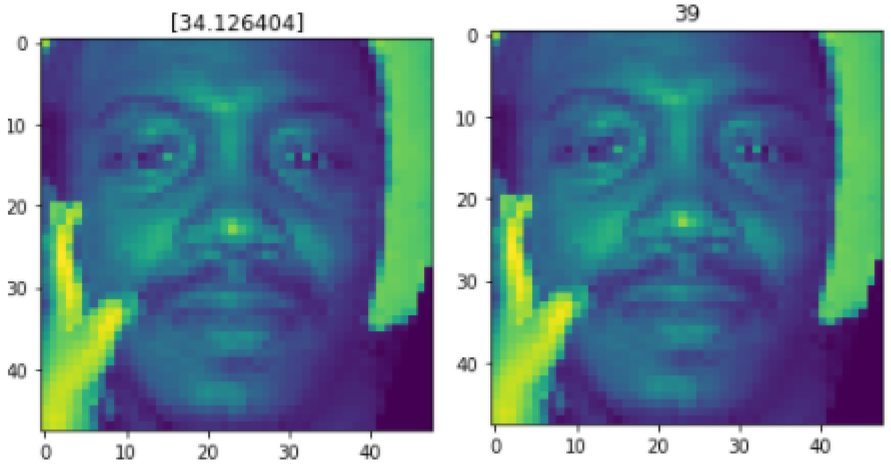
The training and testing data have the same loss.

We evaluated our model's validation "mae" on the test data to estimate the generalization error before deploying the model to production.

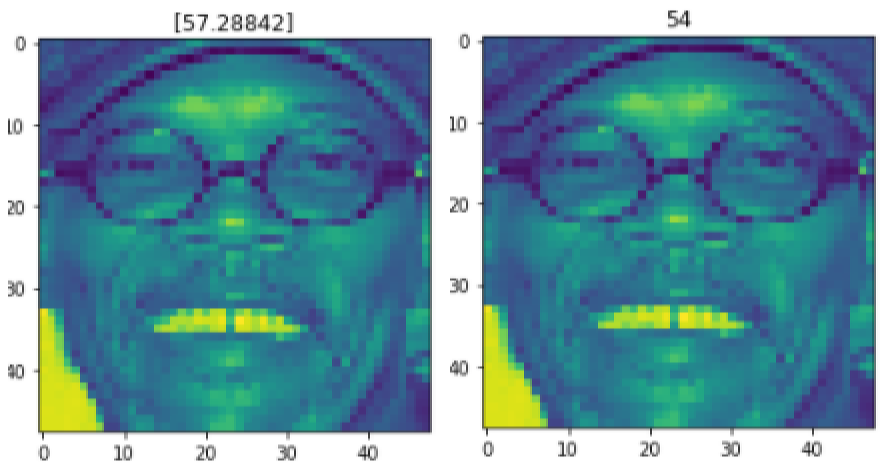
```
Model.evaluate(X_test, y_test)
```

```
149/149 [==============================] - 3s 21ms/step - loss: 155.2118 - mae: 9.4314
[155.21182250976562, 9.431405067443848]
```

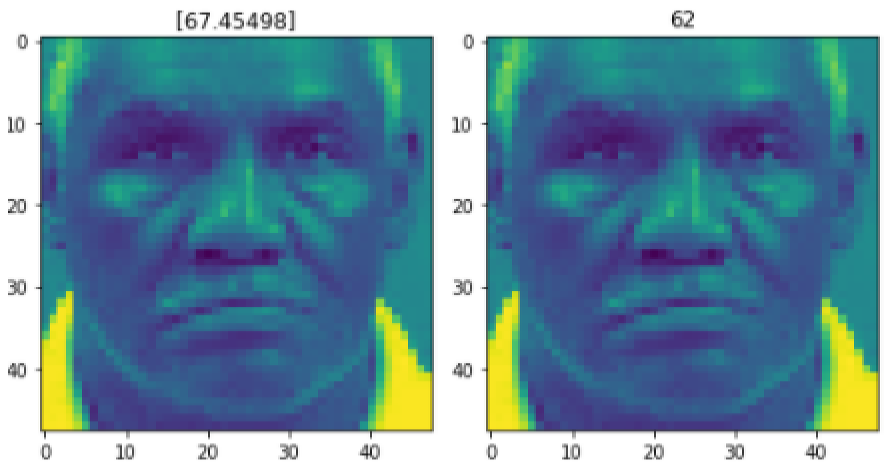
We finally predicted the age of few people in our dataset using our model and compared it with their real age.



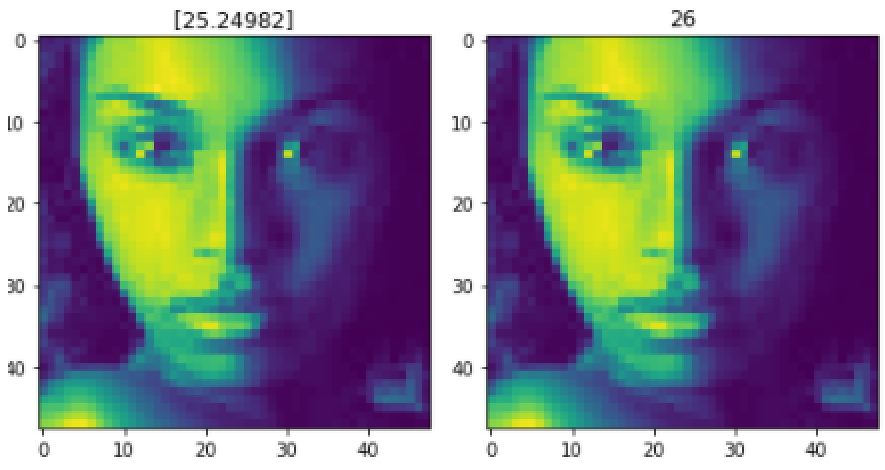
Here our model predicted 34 years old while his real age is 39.



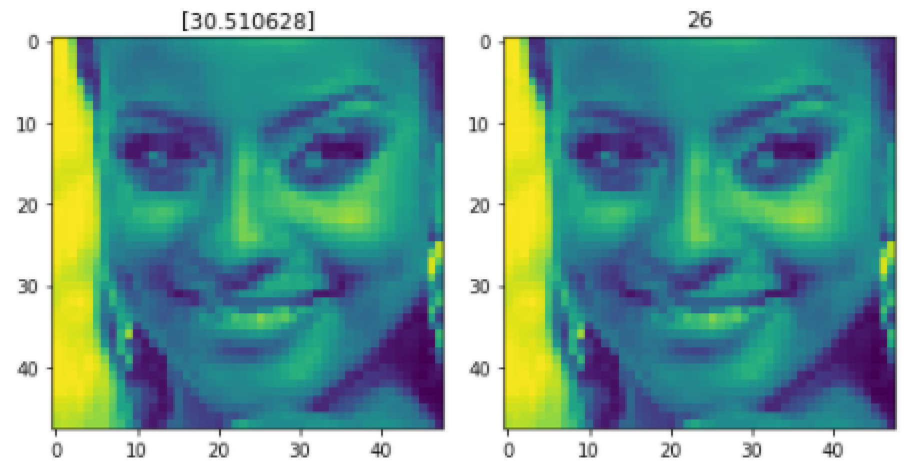
This man is 54 years old but our model predicted 57.



This man is 62 years old but our model predicted 67.



This woman is 26 years old but our model predicted 25.



This woman is 26 years old but our model predicted 30.

VI- NEXT STEPS

To have a better performance metrics we can use the Transfer learning technique. Transfer learning is when we build a new model on top of a pre-trained model. Another challenge of mine is to convert the picture we will take into the same nature as the one that was given to us in the dataset.

We will be able to put our analysis into a software product where we will be able to take people's pictures and the software will output their age. We can use the internet to create a database where we will collect only black people's photos and corresponding age since our project is targeted towards africans. That will help us add more data beneath our model in order to increase the prediction and update our model.