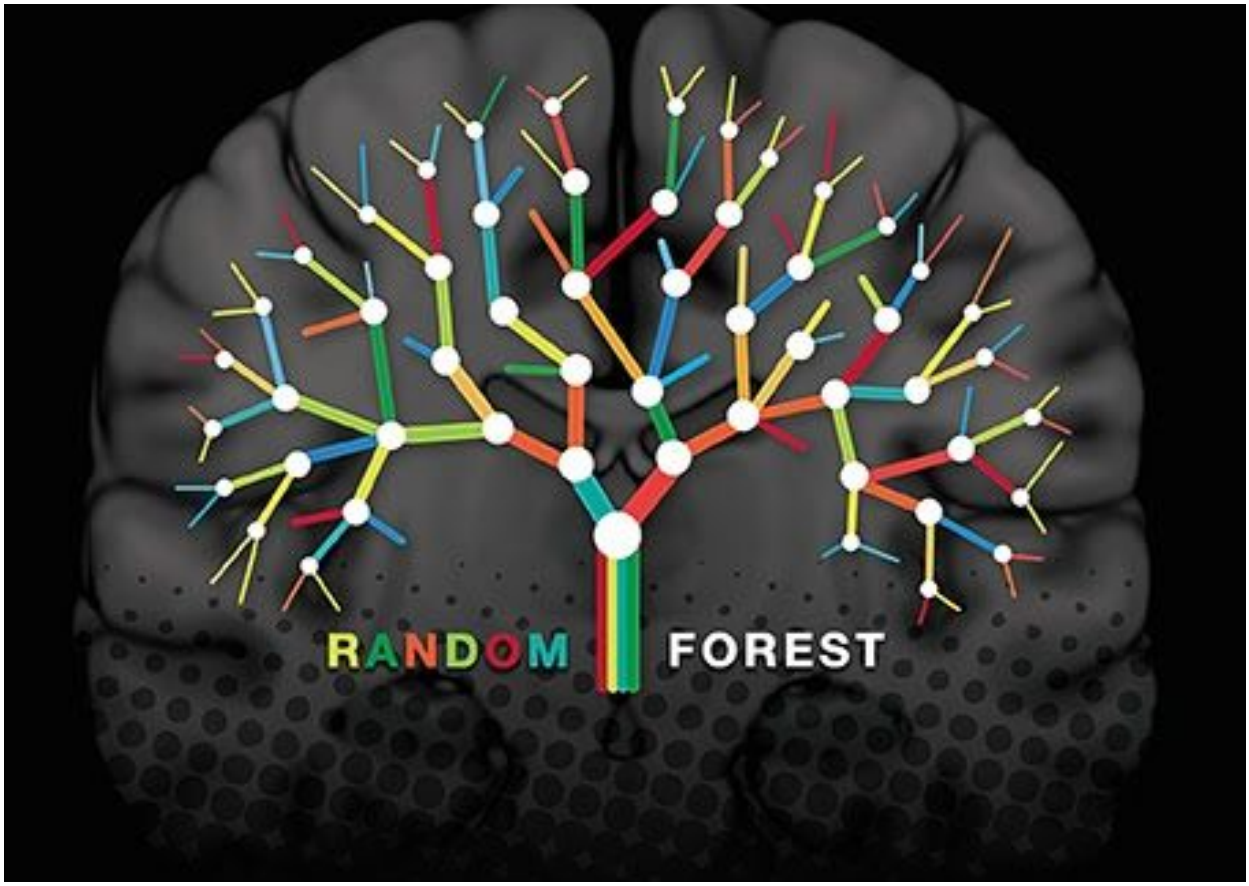
ML workflow project



Santo Plaia - 1611683

12/2020

Big Data for Official Statistics - Data Science

1. Introduction

This project aims to predict the age of a person given a photo of its face using, in this first part, with some traditional machine learning framework.

Since in other courses I have used many times traditional ML with "classical dataset" (i.e. a list of features for each sample) I decided to choose a different type of dataset that consists of only pictures of people's faces with the associated age.

2. The Dataset

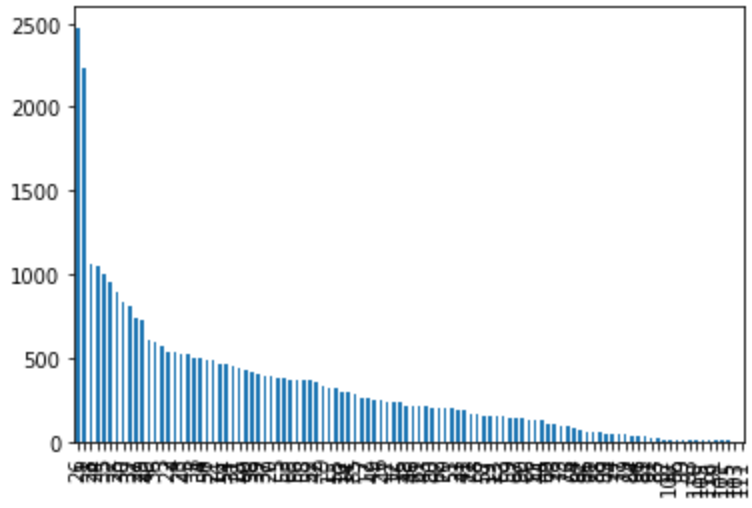
About the dataset I searched online for an appropriate dataset and I found two datasets that was very suitable for my aims:

Name	Dimension of the Dataset (num. of sample)	Dimension of the pics (pixel width x pixel height)	Link
Facial-age (Kaggle)	9778	200px*200px	https://www.kaggle.com/frabbisw/facial-age
UTK Face	23.708	200px*200px	https://susanqq.github.io/UTKFace/
Union	33.486	200px*200px	

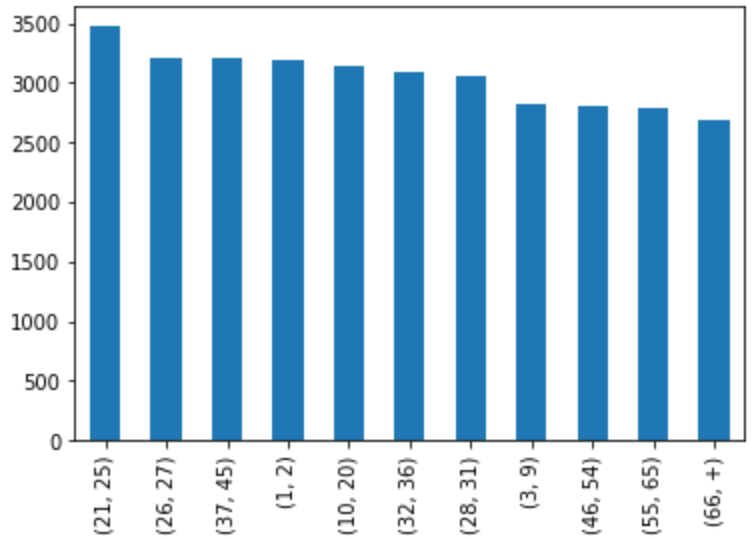
A preliminary work was to join this two dataset obtaining one single dataset of pictures in the format: age_number.jpg; Starting from this collection of pictures I created a simple csv file containing only the *file name* and the *age* of the person in the picture. This was my starting point.

3. Data cleaning and reshaping

Since the distribution of the ages are not uniform (see picture 1), I decided to divide the ages into classes of age in order to obtain a better balanced number of people for each class (see picture 2).



picture 1 - unbalanced ages



picture 2 - balanced ages

4. Preprocessing

I have to preprocess two types of data: the information related to the age for each picture and the pictures.

a. Informations

Since my dataset is made by images for which I have all the necessary information (the associated age), my dataset does not contain any missing values (see picture 3).

Anyway after the grupping of the ages into classes, my dataset contains a string indicating the age range in which the person belongs (see picture 4). For this reason I

decided to encode this feature obtaining a dataset suitable for my project (see picture 5).

```
1 faces.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33486 entries, 0 to 33485
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   file    33486 non-null  object
 1   age     33486 non-null  object
dtypes: object(2)
memory usage: 523.3+ KB
```

picture 3 - informations

```
[ ]  1 faces=pd.read_csv(BASE_PATH + 'faces.csv')
     2 faces.head()
```

	file	age
0	48_65.jpg	(46, 54)
1	28_935.jpg	(28, 31)
2	4_294.jpg	(3, 9)
3	81_8.jpg	(66, +)
4	10_129.jpg	(10, 20)

picture 4 - age ranges

```
[ ]  1 faces.head()
```

	file	age
0	48_65.jpg	8
1	28_935.jpg	4
2	4_294.jpg	5
3	81_8.jpg	10
4	10_129.jpg	1

picture 5 - age encoding

b. Images

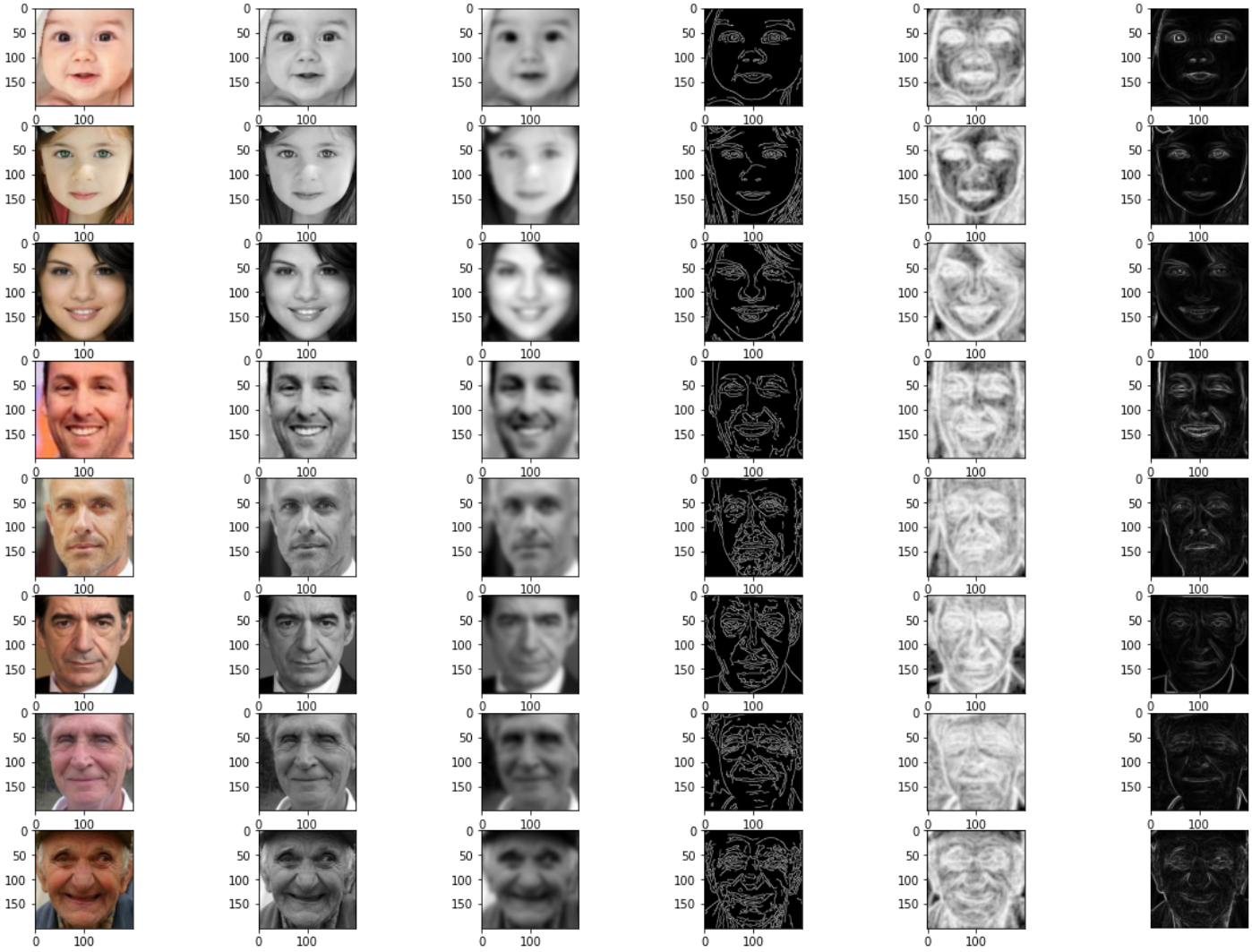
Since in this first part of my project I have to use traditional ML, I must reduce the dimensions of my dataset (33.486 samples x 200px x 200px) to a more simple dataset consisting only in a 33.486 samples and for each one of those I can have a list of features.

My first idea was to flatten each picture but I thought that this can't work well due to the low significance of each pixel.

I decided to divide each picture into sections/square of side 10 pixel and compute for each section the mean and the standard deviation. This approach works more well than the previous.

Thanks to the professor. Scardapane I understood that a group of pictures of a person having the same age are considered different just for the difference of the color of the pixels composing the pictures. For this reason, and under the suggestion of Vamsi, I decided to transform each image from RGB into GrayScale and look for some filters in order to transform my images into something in which the differences of the pixels are less significant. I took a picture for each age range; then I chose a set of filters and I applied them to the photos. The filters are: Gaussian, Canny, Entropy and Sobel.

The result is shown in picture 6 (Original, GrayScale, Gaussian, Canny, Entropy, Sobel):



As we can see, the better filter is the canny one because as the age increases, the number of lines in the transformed picture increases. My hope is that this can help my model to better learn the age of a person.

Also thanks to the professor. Scardapane I decide to use an augmenting technique flipping left-right each image.

So for each picture, my preprocessing is:

1. Read the picture;
2. Flip it (augment);
3. Convert them from RGB to GrayScale;
4. Transform them from GrayScale to Canny;
5. Divide them into sections and
6. for each section take mean and standard deviation.

```
[ ]  1 features
```

	sec1_mean	sec1_std	sec2_mean	sec2_std	sec3_mean	sec3_std	sec4_mean	sec4_std
0	0.19	0.392301	0.00	0.000000	0.00	0.000000	0.10	0.300000
1	0.00	0.000000	0.00	0.000000	0.00	0.000000	0.00	0.000000
2	0.08	0.271293	0.00	0.000000	0.23	0.420833	0.03	0.170587
3	0.10	0.300000	0.09	0.286182	0.00	0.000000	0.09	0.286182
4	0.19	0.392301	0.10	0.300000	0.00	0.000000	0.00	0.000000
...	...	...	...	...	...	...	...	...
66965	0.08	0.271293	0.18	0.384187	0.11	0.312890	0.03	0.170587
66966	0.00	0.000000	0.01	0.099499	0.19	0.392301	0.06	0.237487
66967	0.21	0.407308	0.07	0.255147	0.00	0.000000	0.00	0.000000
66968	0.12	0.324962	0.11	0.312890	0.05	0.217945	0.08	0.271293
66969	0.00	0.000000	0.10	0.300000	0.00	0.000000	0.08	0.271293

66970 rows × 801 columns

```
[ ]  1 features.to_csv(BASE_PATH+'/canny_features_extraction.csv', index=False)
```

After that, the dataset was standardized using a StandardScaler object.

7. Models

For what concern the model I decided to try many model; in particular:

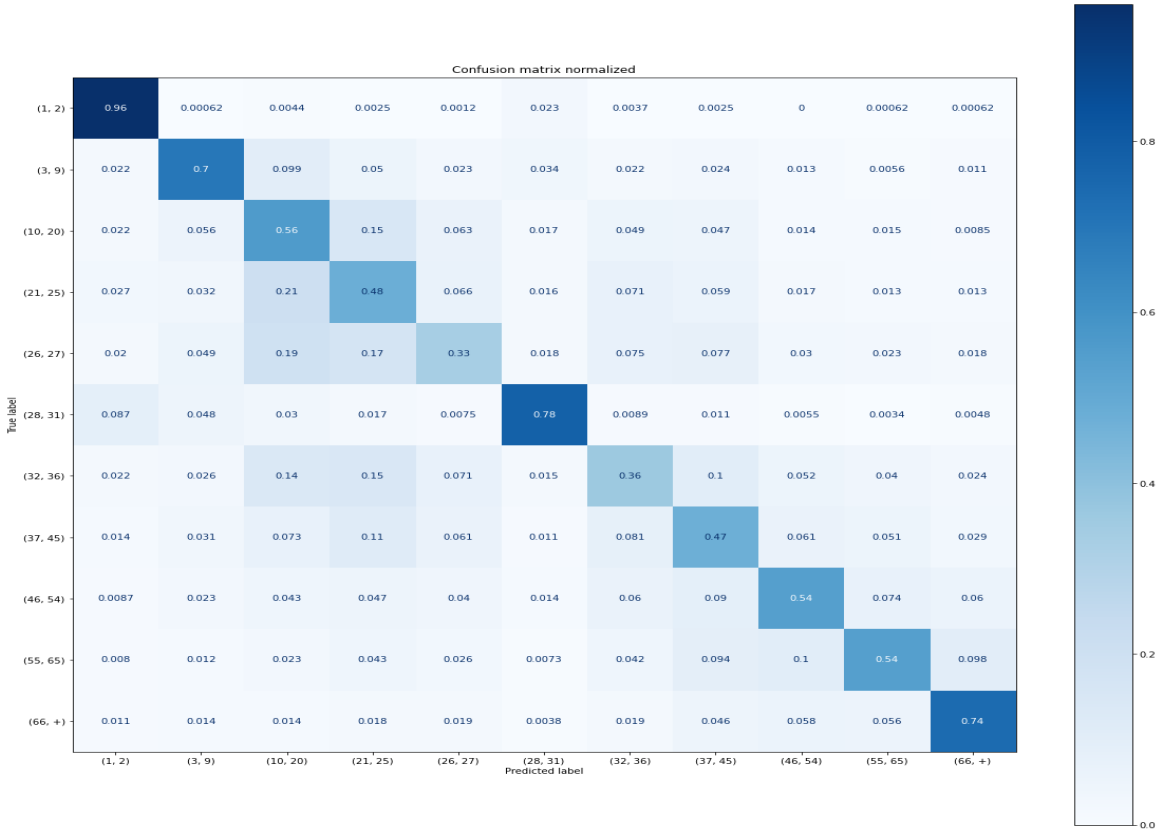
- Logistic Regression
- Random Forest
- Stochastic Gradient Descent
- KNN
- Gaussian NB
- Perceptron
- Linear SVC
- Decision Tree
- SVC

Among these models the best two are: Random Forest and SVC. For these two models I used a Grid Search with 5 cross validation in order to find the best parameter and after comparing the results of the train and test accuracy the overall best model is the Random Forest.

8. Evaluation

About the random Forest, at the end of the parameters tuning the train accuracy was: 0.99605 while the test accuracy was: 0.58902. It's clear that my model was a bit overfitted and for this reason I tried to apply for a PCA but the result was very very poor.

In order to evaluate my model I started with a confusion matrix (picture 8) to understand if there are some mistake patterns.



After the confusion matrix, in order to better understand I investigated the precision, recall and f1-score.

```
[ ]  1 print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0.0	0.81	0.96	0.88	1604
1.0	0.71	0.70	0.71	1612
2.0	0.44	0.56	0.49	1761
3.0	0.39	0.48	0.43	1568
4.0	0.47	0.33	0.39	1579
5.0	0.82	0.78	0.80	1462
6.0	0.45	0.36	0.40	1514
7.0	0.48	0.47	0.48	1600
8.0	0.59	0.54	0.56	1372
9.0	0.64	0.54	0.59	1368
10.0	0.71	0.74	0.73	1303
accuracy			0.59	16743
macro avg	0.59	0.59	0.59	16743
weighted avg	0.59	0.59	0.58	16743

9. My application

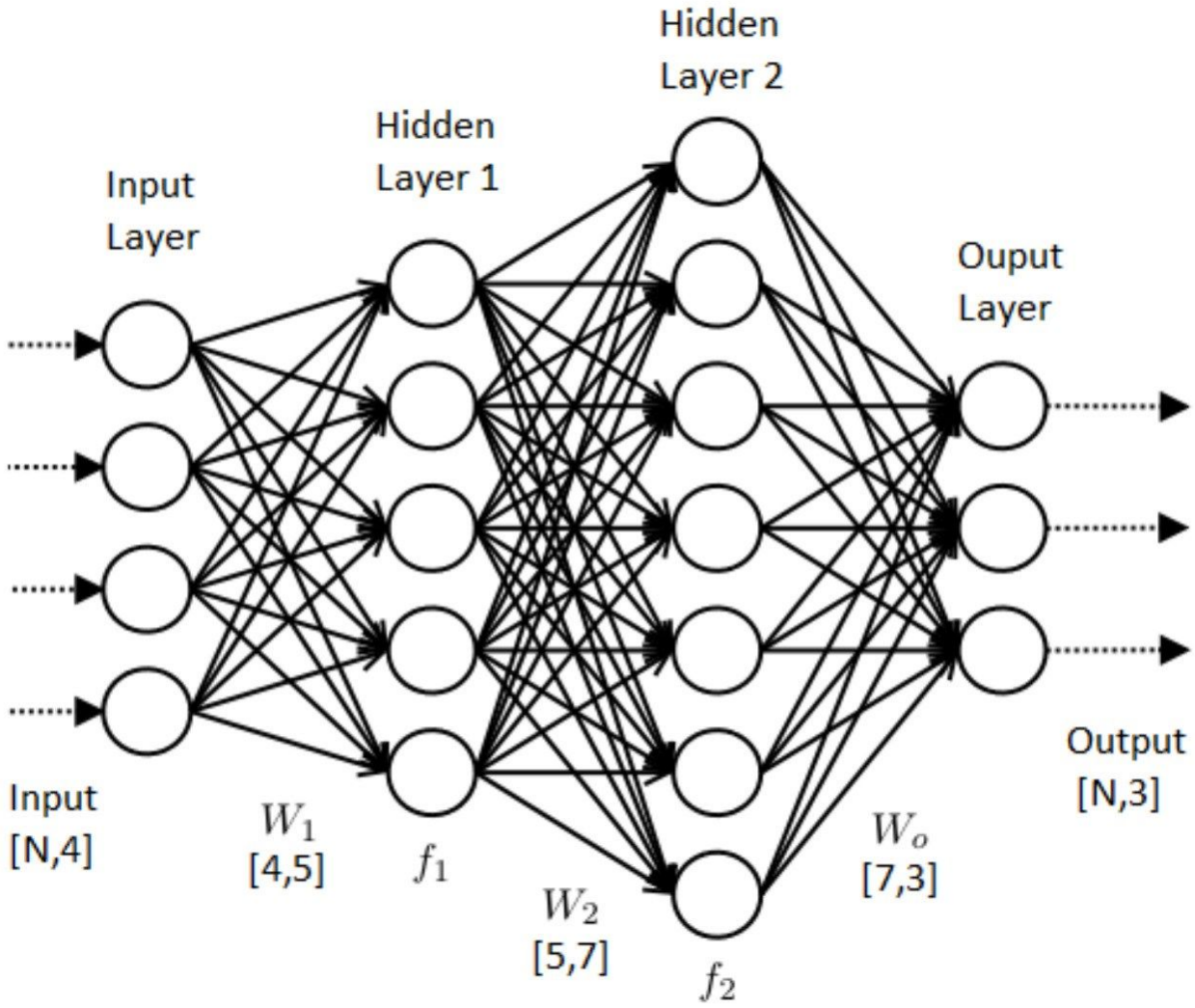
At the following link you can find the first version of my application. You can upload a picture from your computer (using the button “Browse”) and then detect your age (using the button “Upload & detect”)

https://santiagosagedetection.herokuapp.com/

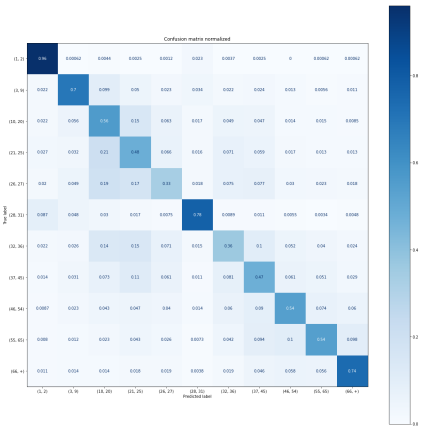
The code behind my application is made by the clone of an existing repository (https://github.com/fcakyon/face-detection-app-tutorial); this choice was made after five version of applications that perfectly run on my local machine but doesn't work properly on heroku. For this reason I decided to find and modify an existing repository to adapt it to my project.

Final Project

A Convolutional Neural Network



1. Preprocess and useful functions



Taking inspiration from the confusion matrix of the ML project, as you can see some ages range are a little bit misclassified.
For this reason I decided to merge some of these ranges. Of course these ranges were encoded prior to use for the model.

Among functions I used, the most important for my model was the augment function in which randomly each picture is flipped left right; this can help my model to better learn looking at the same picture transformed.

The last thing before build my model was the split of my dataset in tree part, the major one is the train part that my model used to learn, the second one is the test part in order to test my model, and the third one is the validation part in order to evaluate the model during the training.

My dataset is not so small and so it is not possible to load all the entries in memory at the same time, for this reason I decided to create and build my model in order to load not the whole dataset, but only batch of my dataset of 32 entries per batch.
Even if my dataset is now divided into batches of 32 samples, they are not so easy to load into the main memory and so I decided to use a prefetch function with an AUTOTUNE parameter in order to try to reduce the latency from the moment in witch my model need data and the time in which data are stored in memory.

2. Model

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 198, 198, 32)      320
_________________________________________________________________
average_pooling2d (AveragePo (None, 99, 99, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 97, 97, 64)        18496
_________________________________________________________________
average_pooling2d_1 (Average (None, 48, 48, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 46, 46, 128)       73856
_________________________________________________________________
average_pooling2d_2 (Average (None, 23, 23, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 21, 21, 256)       295168
_________________________________________________________________
average_pooling2d_3 (Average (None, 10, 10, 256)       0
_________________________________________________________________
global_average_pooling2d (Gl (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 132)               33924
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 931
=================================================================
Total params: 422,695
Trainable params: 422,695
Non-trainable params: 0
```

About the model, since in the ML project I used some classical ML algorithms I decided to use the previous model as baseline for this second part of the project. The aim of my work remains the same but the model changes radically.
In this second part I decided to build a Convolutional neural network using Tensorflow and Keras.

About the structure of my CNN I took inspiration from many famous networks for images; inside my network there are 4 kind of layers:

1. a convolutional one that do a spatial convolution over the images in order to extract features from images appling a number of filter of dimension 3 with a relu activation function;
2. Another layer is the average pooling that splitting the images into sections of 2 times 2 pixels and compute for each the average of 4 pixels reducing the

dimensions of the images;
3. Third layer used is the global average pooling that similar to the previous compute an average over all pixels summarizing the presents of a feature;
4. Last type of layer is a classical dense layer that is a fully connected layer in order to reduce the dimension of the network.

At the end, my network is composed of 5 blocks:
The top four blocks are composed of a convolutional layer and an average pooling layer: in order to find and extract features from images and reduce the dimension of the pictures.
Is important to note that the number of filters applied is inversely proportional to the dimension of the picture: as the picture becomes smaller due to the average pooling, the number of filters increase.

The last block of my network is composed by a global pooling to summarize features in the pictures and 2 dense layers in order to reduce the output of the network to the number of classes, in this case 7 ages range.

The main difference, roughly speaking, between a traditional Machine learning and a Neural network is the presence of some activation functions that introduce in the model a nonlinear operation and allow us to learn the nonlinear component beyond the events we are trying to understand.

The relu function is the most common since it works very similar to neurons and for this reason is a good choice for the inside layers.
For what concerns the last layer, a soft max function is a better choice because this function not only maps the input into a range from 0 to 1 but it maps all inputs into a probability space giving in output the probability of a sample to belong to a certain class, in this case to a certain age range.

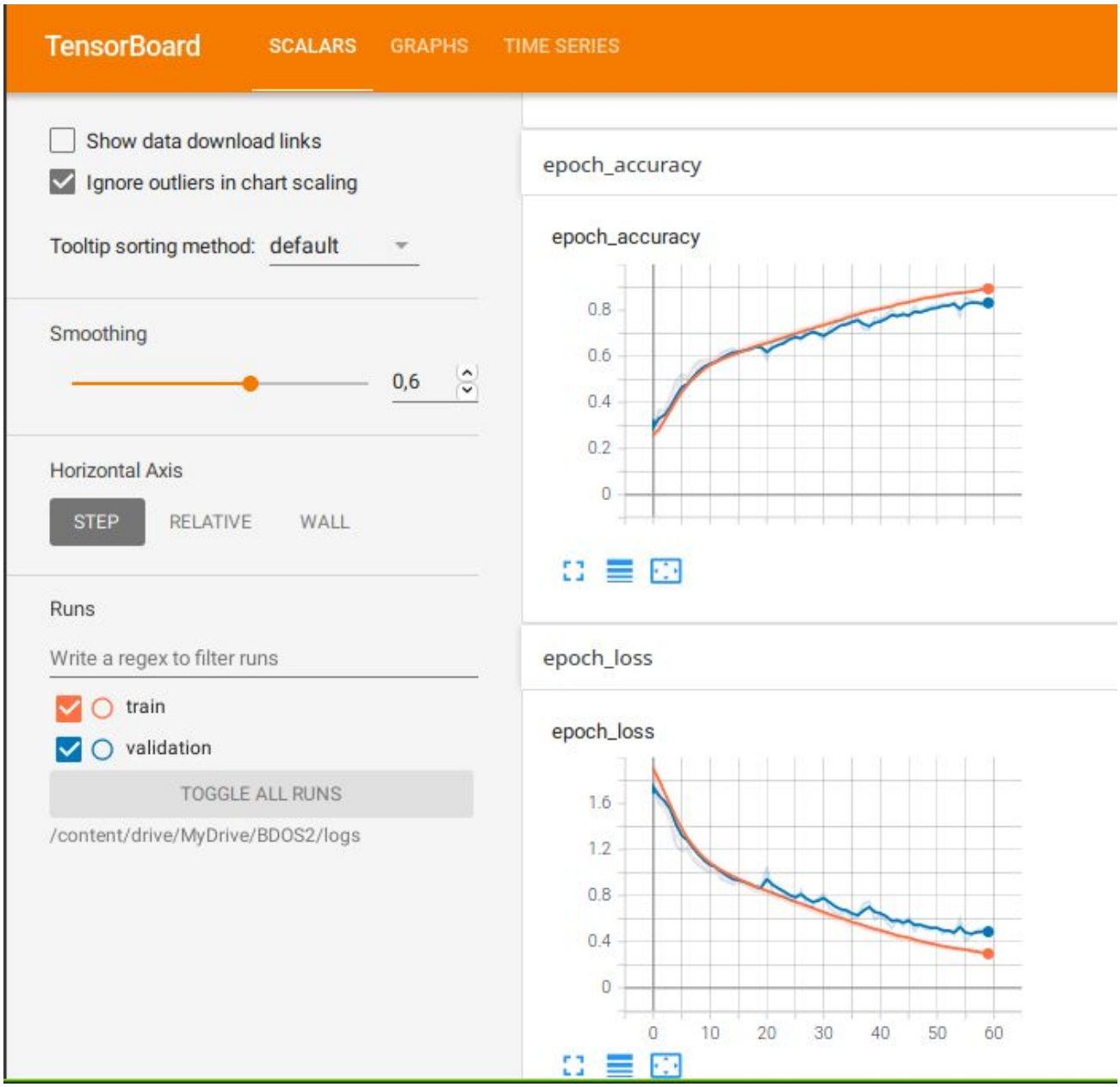
3. Model (Compile)

My model is not simply the CNN but after that I choose other components in order to evaluate, understand and monitor my model.

First of all as a loss function I choose the sparse categorical cross entropy since my target is an integer and since I have to classify data into classes mutually exclusive.
Instead of the classical gradient descent I decided to use an extension of this latter using the adam optimizer.
For what concerns the metrics for evaluating the model I choose the classic accuracy in order to measure the number of correct classifieds.

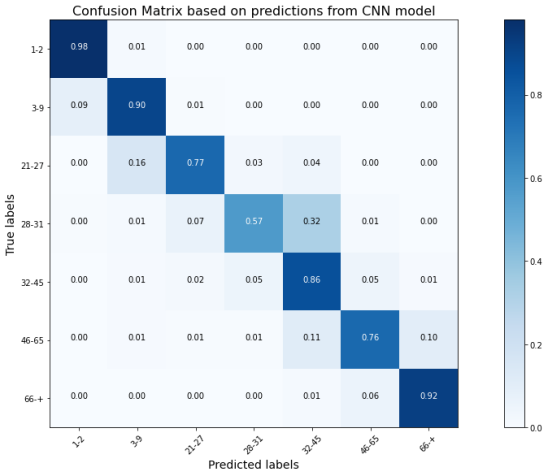
Since my dataset was split in batches the accuracy of my model can be unstable over the epochs of my training part and for this reason I decided to use a checkpoint object in order to save the best weights for the validation accuracy.

Finally, in order to visualize the trend of my learning part I used a Tensorboard and we can see it in this picture.



4. Evaluation

For what concerns the evaluation of my model, even in this case I decided to use a confusion matrix in order to highlight some misclassified patterns. As you can see the diagonal contains the largest number of samples, but we can see there are some small misclassified from th and the adjacent ranges.

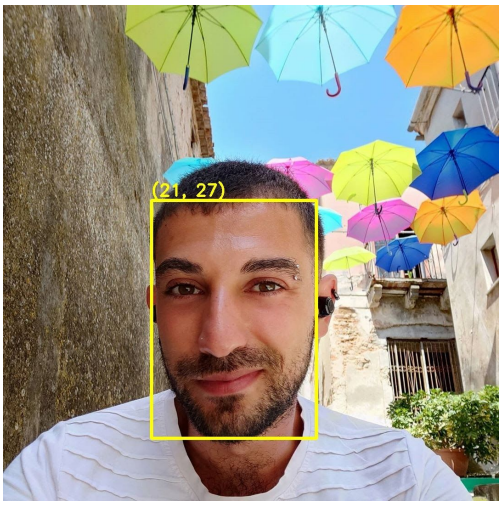


5. App

At the following link you can find the application (the same of the ML project) with the CNN incorporated inside: [Palaia Age Prediction App]

About the app it was built using Heroku and Flask. Heroku can host my repository and manage the part related to the real-time deployment of the app while Flask manage the creation of the frontend and the backend processing.
In the app, a client can upload a picture and see it with the age prediction.

For the frontend, Flask was set to manage 3 pages: a root page, an upload page in order to upload a picture and a detected page in order to display the picture with the prediction, as you can see.



A special thanks goes to Fatih from the Bilkent university for the skeleton of my app.
I developed 3 different apps for my project but no one of those works correctly on heroku due to the difficulty of obtaining the client camera control from the server. For this reason I decided to take a workin app and rearrange it in order to obtain a working app for my project.

Here the link to the original app:
[face-detection-app-tutorial]

6. Video

For what concern the apps developed totally from scratch by me, we can see this small videos in which my app predict the age directly on the video streaming:

[First Video]

[Second Video]

These apps are created using

- Flask: for the logic behind the app
- WebCamVideoStream: to capture the computer camera
- Open-cv.CascadeClassifier: to find out the face inside the video stream

Conclusions

After developing two different models for the same task with the same dataset I can make some conclusion: first of all I can claim that a neural network is better than a classical machine learning algorithm for two reason: first - transforming each image into a vector of features leads to the lost of many information like the position of the pixels and other information about the context of each pixel.
After that we have to consider also the presence of a non-linear component inside the network that helps the model to better understand.