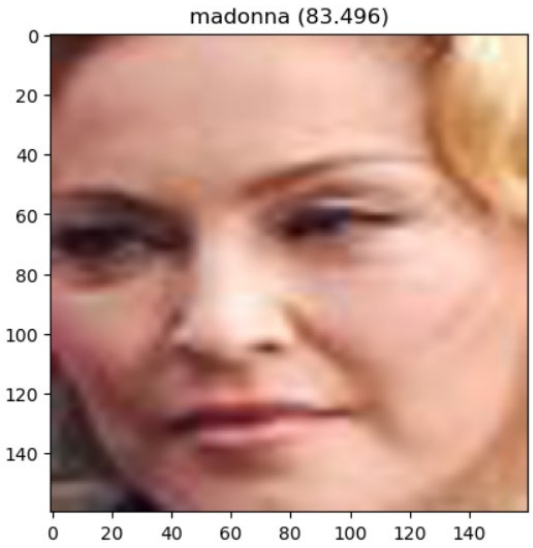
Session 3: detecting faces with FaceNet



In this assignment, you will have to write the pipeline in order to apply a FaceNet model on faces of a dataset and classify the face as the right person. This code will be extendable to new faces without the need of retraining the FaceNet model. However, as discussed in the theory, the more new and unfamiliar faces you add, the less the model will be able to distinguish between the faces. In that case, you can retrain FaceNet, but that is out of scope for this assignment. Execute the following steps:

1. Download the **5_celebrity_faces.zip** dataset on Leho. **Note:** while unzipping the folder on a Windows machine, you might encounter an error due to the path length being too long. You can solve this by extracting the folder closer to the root dir (C: for example). You should not encounter this problem on a linux machine.
2. Apply the **MTCNN** model (or Multi-Task Cascaded Convolutional Neural Network) on the dataset. This model will cut out the faces from the images and therefore does face detection. It does not do face recognition, that is why we need FaceNet. You can install MTCNN in your environment with 'pip install mtcnn'. Remember to preprocess your images in the right way before feeding them to the network. Store the detected faces as size 160*160*3 images.
3. Convert your detected faces to an embedding using FaceNet. Download the **facenet_pretrained.zip** folder on leho. This contains the model and weights of a FaceNet model with Inception ResNet v1 backbone pretrained on MS-Celeb-1M dataset. You can find the code on https://github.com/nyoki-mtl/keras-facenet but you will not need it for this assignment. Preprocess the faces before feeding them to FaceNet. The model requires standardized images (subtract mean and divide by std) with size (1,160,160,3). The output of the model will be an embedding of size 128. **Note:** the FaceNet model has been trained with an older version of TensorFlow. If you want it to work, you can use the following package versions:
    - TensorFlow 2.0
    - Keras 2.3.1
    - Tensorflow-estimator 2.0

- h5py 2.10
- python 3.7
- mtcnn 0.1.1

4. Use the embedded training images to train a linear support vector machine. Normalize the embeddings before training and testing on them. Determine the accuracy score on the training and test set.
5. Choose a random image from the test set. Predict what person it is and it's probability. Plot the face together with the probability.
6. **Optionally**: if you want, you can now also apply this pipeline on your own face by adding pictures of yourself in the training and test set, applying MTCNN and FaceNet and retraining your SVM classifier.