Task 1: Skin Detection Part 2 (practical)

In the last exercise, we were creating a data set for skin detection to localize faces in images. To obtain a full data set, download the skin segmentation data set from the UCI Machine Learning Repository:
https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation

Use a standard framework for machine learning:

- **scikit-learn** (http://scikit-learn.org) is an advanced machine learning library for Python. It contains many libraries for classification and regression.
- **TensorFlow** (https://www.tensorflow.org) is advanced machine learning and neural networking library with a front end for Python.
- …many other libraries, so pick your preferred option.

a) Prepare data: either use the result from Exercise 4 or download the data set from UCI Machine Learning Repository: note that the data contains about 50k skin values, and 195k non-skin values. Each line contains 4 values: blue, green, red, and a label (1: skin, 2: non-skin).

b) Apply the data set to a classification or neural network approach. Train it with your data and assess the result with an independent test set.

c) Apply the resulting prediction model on an image with faces. Filter the image such that skin pixel appear white and non-skin pixel are black. How well does your method work with real images?

Task 2: MNIST Challenge (practical)

We will use the TensorFlow library and apply it to the MNIST Challenge, i.e., recognition of handwriten digits. The exercise is essentially running the tutorial on the TensorFlow page to understand how a deep learning network is applied to a real life example.

We will use the following library for this exercise:

- **TensorFlow** (https://www.tensorflow.org) is advanced machine learning and neural networking library with a front end for Python.

a) MNIST is a data set with handwritten digits. The data set is already available in the tensorflow package, split into training and test data set to provide a normalized benchmark environment. Follow the tutorial https://www.tensorflow.org/get_started/mnist/pros

b) The tutorial starts with a simple network with a single layer fully connecting the 28x28 pixels to 10 output values using softmax. The cross-entropy function is used. Extend this model with additional hidden layers and observe how the model improves.

c) The tutorial also contains a more complex model (the training can run several minutes to an hour depending on your machine, so be mindful about the number of training iterations). It uses convolution, pooling, dropouts, and fully connected layers. It also employs a more sophisticated training algorithm (ADAM Optimizer). Play through the tutorial, lookup the functions to understand how the network works. Try to adjust the network in some way and see how it evolves. Why not implement an inception layer like the one from GoogleLeNet? Can you beat the initial 99.2% accuracy?