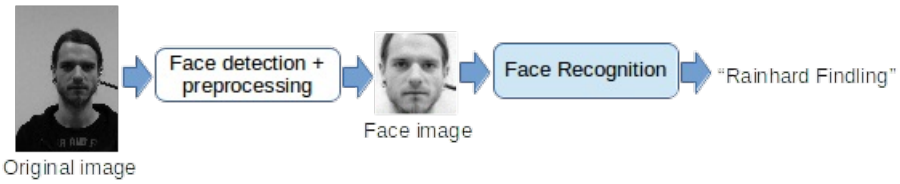
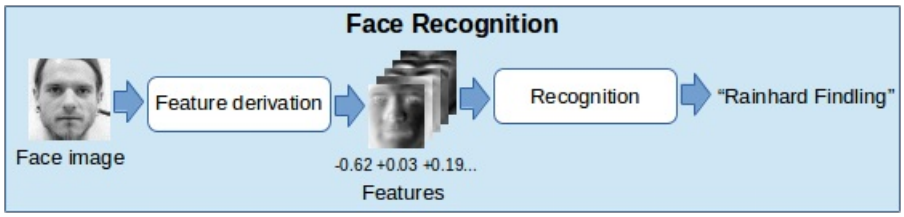
Face Recognition





Task

- Goal is to build (a) successful face recognition model(s). The model should learn to distinguish between N people from face images, then being able to decide for new face images which of the N participants shown. Load the data, derive features and evaluate different models. Try to achieve the best possible performance by changing your feature derivation (always use proper data partitioning):
  - Take some of the original pixel features of your data that lie new to each other (e.g. first 100) and do a correlation plot for them. What do you see?
  - Feature derivation likely is a multi-step-process. You can use other feature derivation approaches than those discussed in the lecture too -- there are many of them. Use "whatever floats your boat", but be sure to understand your own toolchain. E.g. try changing details, like differently sized input images, more/less/different features, and compare differences in results.
  - What is the best approach (features, model, ...) you can come up with to successfully distinguish people? What do you think of the data/the results? Chose appropriate metrics to underline your statements.
- After solving the above points, answer this question: what happens if we build our model from such a data set, then somebody not part of it uses a system where the model is deployed? What could you do about it/how could such systems possibly work?

Dataset

- The *frontal-only* face images from the Panshot Face Database should be used. Details about the database structure and file naming can be found at https://usmile.at/downloads#panshot_face_database.
- Shortcut 1: there is a already face-detected and normalized face image dataset available at http://dev.usmile.at/authentication/public-face-db/2013-01/dsr-preprocessed-1000x1333_faces_haar_gray_resized150_equalized.7z, which saves you the effort of face detection+segmentation+histogram equalization.
- Shortcut 2: there is a downsized version of the frontal-only part of the dataset above available in elearning.

Hints

- This is the first assignment with a real life problem and dataset.
- Computer vision like tasks (like face recognition) are usually done on grayscaled and normalized images - no immediate need for color.
- Each pixel of a face image can be treated as feature (e.g. 150x150 pixels = 22500 features = 22500 dimensions, 50x50 pixels = 2500 features, ...). Representation of face samples in a csv file/data frame would be the same as for other problems: 1 sample (=1 face image) per line, with each line having e.g. 10000 or 2500 features = pixels.
- Images should be loaded using a png library (e.g. `readPNG()` from `png`). Loading should be done using a vectorized operation and result in one dataframe that contains all pixels of one image per line as stated above - similar to `ldply(dir(..., full.names = T, pattern = 'png'), function(f) t(as.numeric(readPNG(f))))`. *Bringing data into the form of such a dataframe is essential.*
- Due to the high dimensionality, downsizing images and using dimensionality reduction is recommended. Additional feature derivation will prove useful too:
  - Hint: if you use PCA you can transform data into PCA space, leave out the less important components

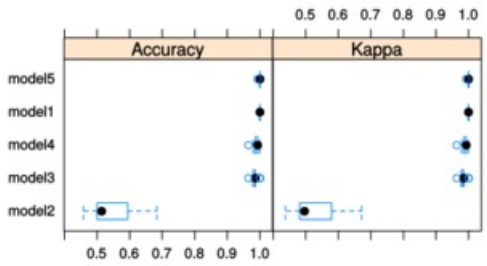
(dimensionality reduction), then transfer your samples back into the original space to see for the effect of dimensionality reduction. These will be visible as artifacts (blurry etc). If you can't distinguish people after transforming dimensionality reduced data back to original space, you probably overdid the job. This piece of *human verifiable* information can be very helpful in finding out a good amount of dimensions to use in such cases.
    - Other transformations could be useful too (e.g. decomposing faces into Fourier components, wavelets, ...).
- In case you struggle with RAM: downsize images, reduce dimensions, reduce parallelization -- and apply clever feature transformation before building the model.
- For transformations influenced by data distribution (e.g. PCA) remember to only transform training data when training your model, then to apply the same transformation on cross validation/test data too. With `R` and `caret` you can e.g. use the `preProcess` parameter of `caret::train` for this to be done correctly.
- Possible steps: downsize images, load images and labels, do data partitioning, derive features (think of if you need to do it in `caret::train` with `preProcess(...)`), train+evaluate models, chose model to use, obtain test performance.

Hand-in

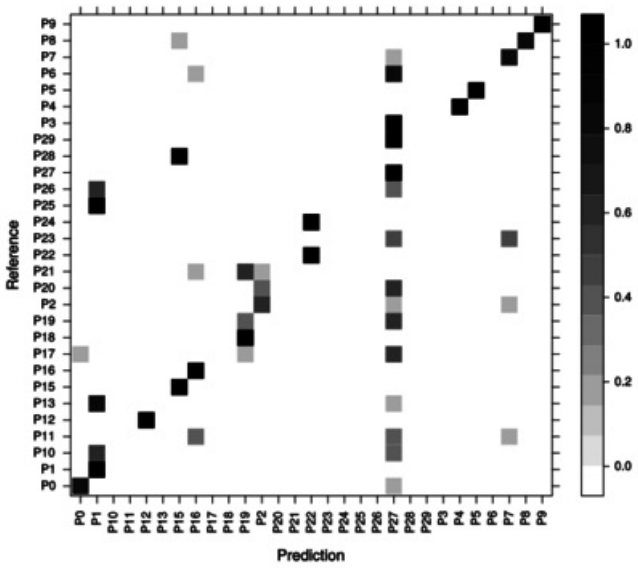
- You can chose tools as you like and can span your toolchain across multiple platforms/languages. For R and alike parts hand in: source code + exported model object stored with `saveRDS` as `RData` file; for Weka parts hand in: detailed description of steps done and exported model. If models are too big for uploading, simply skip them in your export.
- In the report: document everything so that **your system could be rebuilt from using the report only**:
    - Flowchart/block diagram of the toolchain you use, containing each step from the original data to model training and evaluation performance testing.
    - State clearly how you split data/do data partitioning.
    - State clearly which feature derivation/transformations with which parameters you apply.
    - State and compare performance of different models/approaches.
    - State which model you would chose for a real world application, why you would do so, and its performance (confusion matrix, ...).
    - Don't forget to state your thoughts/findings on your data at each point in the toolchain (why you apply transformations, what you see, etc).
- This assignment will be credited by how creative you were, how precise you performed and documented the task, and of course by your findings and solutions. Therefore, the report is highly important for this assignment: everything you want reviewers to *understand* needs to be stated in it.

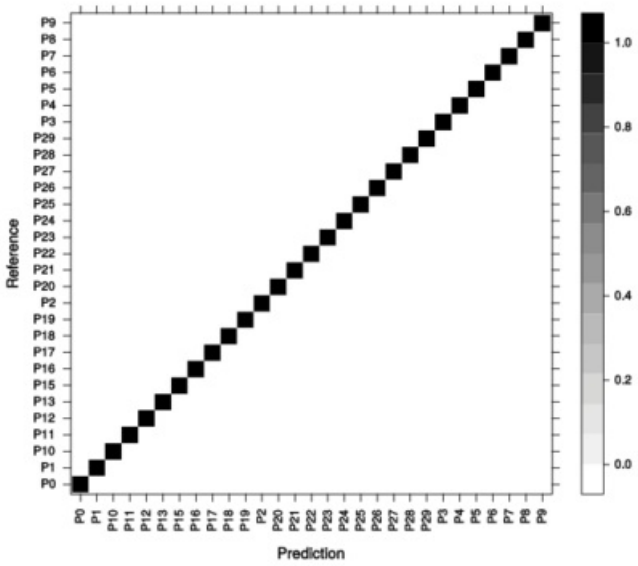
Examples

Visualizing your results in a similar way will be beneficial. Some "good" and one "bad" model:



"Bad" and "good" confusion matrix:


Reference
Prediction
P0 P1 P10 P11 P12 P13 P15 P16 P17 P18 P19 P2 P20 P21 P22 P23 P24 P25 P26 P27 P28 P29 P3 P4 P5 P6 P7 P8 P9
1.0
0.8
0.6
0.4
0.2
0.0


Reference
Prediction
P0 P1 P10 P11 P12 P13 P15 P16 P17 P18 P19 P2 P20 P21 P22 P23 P24 P25 P26 P27 P28 P29 P3 P4 P5 P6 P7 P8 P9
1.0
0.8
0.6
0.4
0.2
0.0