Coding Task

To prepare for the task, please take a look at the latest version of our openSMILE toolkit (v2.3.0 from https://www.audeering.com/opensmile/) and try to understand the basics from the documentation (openSMILE book). Please try to use CMake or another modern build tool to compile openSMILE, and share the build script(s) in a Git repository.

The tasks will involve the use of openSMILE for audio analysis, as well as a machine-learning framework such as Python's scikit-learn library for classifying with GMMs and support vector machines (SVMs), for example. As example audio dataset you should use the Berlin Speech Emotion Database (Emo-DB: http://emodb.bilderbar.info/docu/).

Using these tools and data, please build a simple emotion recognition classifier for and from the Emo-DB recordings. The following steps are necessary:

1. You should run openSMILE's `SMILExtract` tool on each audio file in Emo-DB and serialize or dump the resulting features in a single file, using a generic format such as CSV, JSON, or YAML. Please use the configuration file: `config/IS13_ComParE.conf`. Implement this using a build automation tool such as Gradle, Ant, Rake, Bazel, etc. and share the code in the Git repository. Document your build logic as appropriate, if task or function names are not obvious, and provide a README that explains the prerequisites and provides usage instructions.

   Hint: This configuration file includes other configuration files, which is not yet well documented in the openSMILE book. The command that includes another config file within a config file is `\{filename.conf}`. Please read the config file to find these dependencies. The options for data output can be viewed and modified in the file `config/shared/standard_data_output.conf.inc`. The type of output file is controlled by the commandline options defined in this file (`\cm[...]` - documented in the openSMILE book), i.e., setting one of the options enables the output component.

   Hint: It might be required to convert the audio files of Emo-DB to a format that openSMILE can read, if you get error messages from openSMILE regarding the WAV header format. You can use the tool `sox` for that. Instructions are provided in the openSMILE book.

2. Use the scikit-learn library (or another open-source machine learning framework if you prefer) to build a classifier (SVM and GMM) for the 7 emotion classes (including neutral) of Emo-DB. You should read in the feature files produced in step 1, think of a way of representing the labels (e.g., convert string to numerical indices, or use strings as label names?) and of assigning the correct label (path name in the Emo-DB dataset) to each feature vector (row in the CSV file).

   You should define a training set and a test set for Emo-DB, with a fixed random seed for reproducibility. The sets should be speaker independent, i.e., you should select about 2/3 of the speakers to be in the training set and 1/3 of the speakers to be in the test set. The models (SVM and GMM)

should be trained on the training set feature vectors and applied to the test set feature vectors. On the test set feature vectors you should compute recall rate (accuracy) as evaluation measure and have the script print that.

3. Generate a PDF or HTML report (using LaTeX, R markdown, reStructured Text, AsciiDoctor, or similar) that consumes and presents the results of the previous step. Format accuracy in one or more tables, and visualize it in suitable plots. Ideally, this report should be created automatically using the build automation tool of your choice.

As a deliverable, please provide access to a Git repository that contains all code and documentation to perform the steps above, and provide the report as an assets or webpage. Ideally, everything can be run in a clean sandbox environment such as a stock Ubuntu VM or Docker container. It's perfectly acceptable to keep this private, as long as you share it with us.

With this task we want to see how you solve problems, and how you address unknown issues or missing specifications as well as how you approach software engineering and data science tasks. You don't have to deliver a perfect solution, and in many parts our specification allows a lot of freedom for you to tweak parameters implement certain steps, where we would like to see which way you choose. Even if you are not able to fully solve all tasks in the time available, please try to solve as much as possible, and add comments in your code where things are missing and why, and what you would need to solve to get those issues sorted.