

# Nkululeko 1.0: A Python package to predict speaker characteristics with a high-level interface

Felix Burkhardt<sup>1,2\*</sup> and Bagus Tris Atmaja<sup>3\*</sup>

<sup>1</sup> audEERING GmbH, Germany <sup>2</sup> TU Berlin, Germany <sup>3</sup> National Institute of Advanced Industrial Science and Technology (AIST), Japan \* These authors contributed equally.

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#)
- [Repository](#)
- [Archive](#)

Editor: [Open Journals](#)

## Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

## Summary

Nkululeko (Burkhardt, Wagner, et al., 2022) is open-source software written in Python and hosted on GitHub. It is predominantly a framework for audio-based machine learning explorations without the need to write Python code, and is strongly based on machine learning packages like sklearn (Pedregosa et al., 2011) and pytorch (Chaudhary et al., 2020). The main features are: training and evaluation of labelled speech databases with state-of-the-art machine learning approach and acoustic feature extractors, a live demonstration interface, and the possibility to store databases with predicted labels. Based on this, the framework can also be used to check on bias in databases by exploring correlations of target labels, like, e.g. depression or diagnosis, with predicted, or additionally given, labels like age, gender, signal distortion ratio or mean opinion score.

## Design choices

The program is intended for **novice** people interested in speaker characteristics detection (e.g., emotion, age, and gender) without being proficient in (Python) programming language. Its main target is for **education** and **research** with the main features as follows:

- Finding good combinations of variables, e.g., acoustic features, models (classifier or regressor), feature standardization, augmentation, etc., for speaker characteristics detection (e.g., emotion);
- Characteristics of the database, such as distribution of gender, age, emotion, duration, data size, and so on with their visualization;
- Inference of speaker characteristics from a given audio file or streaming audio (can be said also as “weak” labeling for semi-supervised learning).

Hence, one should be able to use Nkululeko after installing and preparing/downloading their data in the correct format in a single line.

```
$ nkululeko.MODULE_NAME --config CONFIG_FILE.ini
```

## How does it work?

nkululeko is a command line tool written in Python, best used in conjunction with the Visual Studio Code editor (but can be run stand-alone). To use it, a text editor is needed to edit the experiment configuration. You would then run nkululeko like this:

```
$ nkululeko.explore --config conf.ini
```

34 and inspect the results afterward; they are represented as images, texts, and even a fully  
35 automatically compiled PDF report written in latex.

36 nkululeko's data import format is based on a simple CSV formalism, or alternatively, for  
37 a more detailed representation including data schemata, audformat.<sup>1</sup> Basically, to be used  
38 by nkululeko, the data format should include the audio file path and a task-specific label.  
39 Optionally, speaker ID and gender labels help with speech data. An example of a database  
40 labelled with emotion is

```
file, speaker, gender, emotion
x/sample.wav, s1, female, happy
...
```

41 As the main goal of nkululeko is to avoid the need to learn programming, experiments are  
42 specified by means of a configuration file. The functionality is encapsulated by software  
43 *modules* (interfaces) that are to be called on the command line. We list the most important  
44 ones here:

- 45 ▪ **nkululeko**: do machine learning experiments combining features and learners
- 46 ▪ **demo**: demo the current best model on the command line or some files
- 47 ▪ **test**: run the current best model on a specified test set
- 48 ▪ **explore**: perform data exploration (used mainly in this paper)
- 49 ▪ **augment**: augment the current training data. This could also be used to reduce bias  
50 in the data, for example, by adding noise to audio samples that belong to a specific  
51 category.
- 52 ▪ **aug\_train**: augment the training data and train the model with the augmented data.
- 53 ▪ **predict**: predict features like speaker diarization, signal distortion ratio, mean opinion  
54 score, arousal/valence, age/gender (for databases that miss this information), with deep  
55 neural nets models, e.g. as a basis for the *explore* module.
- 56 ▪ **segment**: segment a database based on VAD (voice activity detection)
- 57 ▪ **ensemble**: ensemble several models to improve performance

58 The configuration (INI) file consists of a set of key-value pairs that are organised into several  
59 sections. Almost all keys have default values, so they do not have to be specified.

60 Here is a sample listing of an INI file (`conf.ini`) with a database section:

```
[EXP]
name = explore-androids
[DATA]
databases = ['androids']
androids = /data/androids/androids.csv
target = depression
labels = ['depressed', 'control']
samples_per_speaker = 20
min_length = 2
[PREDICT]
sample_selection = all
targets = ['pesq', 'sdr', 'stoi', 'mos']
[EXPL]
value_counts = [['gender'], ['age'], ['est_sdr'], ['est_pesq'], ['est_mos']]
[REPORT]
latex = androids-report
```

61 As can be seen, some of the values simply contain Python data structures like arrays or  
62 dictionaries. Within this example, an experiment is specified with the name *explore-androids*,

<sup>1</sup><https://audeering.github.io/audformat/>

63 and a result folder with this name will be created, containing all figures and textual results,  
64 including an automatically generated Latex and PDF report on the findings.

65 The *DATA* section sets the location of the database and specifies filters on the sample, in this  
66 case limiting the data to 20 samples per speaker at most and at least 2 seconds long. In this  
67 section, the split sets (training, development, and test) are also specified. There is a special  
68 feature named *balance splits* that lets the user specify criteria that should be used to stratify  
69 the splits, for example, based on signal distortion ratio.

70 With the *predict* module, specific features like, for example, signal distortion ratio or mean  
71 opinion score are to be predicted by deep learning models. The results are then used by a  
72 following call to the *explore* module to check whether these features, as well as some ground  
73 truth features (*age* and *gender*), correlate with the target variable (*depressed* in the given  
74 example) in any way.

75 The *nkululeko* configuration can specify further sections:

- 76     ▪ **FEATS** to specify acoustic features (e.g. *opensmile* (Eyben et al., 2010) or deep learning  
77        embeddings; e.g. *wav2vec 2.0* (Baevski et al., 2020)) that should be used to represent  
78        the audio files.
- 79     ▪ **MODEL** to specify statistical models for regression or classification of audio data.

## 80 Example of usage

81 In the previous section, we have seen how to specify an experiment in an INI file that can  
82 be run with, for instance, *explore* and *segment* modules. Here, we show how to run the  
83 experiment (*nkululeko.nkululeko*) with built-in dataset (Polish Speech Emotions dataset)  
84 from the installation until getting the results.

85 First, novices could clone the GitHub repository of *nkululeko*.

```
$ git clone https://github.com/felixbur/nkululeko.git
$ cd nkululeko
```

86 Then, install *nkululeko* with *pip*. It is recommended that a virtual environment be used to  
87 avoid conflicts with other Python packages.

```
$ python -m venv .env
$ source .env/bin/activate
$ pip install nkululeko
```

88 Next, extract *polish\_speech\_emotions.zip* inside the *nkululeko* data folder (*nkululeko/data/polish*)  
89 with right click regardless of the operating system (or using *unzip* command in the terminal  
90 like below). Then, run the following command in the terminal:

```
$ cd data/polish
$ unzip polish_speech_emotions.zip
$ python3 process_database.py
$ cd ../..
$ nkululeko.nkululeko --config data/polish/exp.ini
```

91 That's it! The results will be stored in the *results/exp\_polish\_os* folder as stated in *exp.ini*.  
92 Below is an example of the debug output of the command:

```
DEBUG: nkululeko: running exp_polish_os from config data/polish/exp.ini,
nkululeko version 0.91.0
...
DEBUG: reporter:
           precision    recall  f1-score   support
```

anger	0.6944	0.8333	0.7576	30
neutral	0.5000	0.4333	0.4643	30
fear	0.6429	0.6000	0.6207	30
accuracy			0.6222	90
macro avg	0.6124	0.6222	0.6142	90
weighted avg	0.6124	0.6222	0.6142	90

```
DEBUG: reporter: labels: ['anger', 'neutral', 'fear']
DEBUG: reporter: result per class (F1 score): [0.758, 0.464, 0.621]
from epoch: 0
DEBUG: experiment: Done, used 7.439 seconds
DONE
```

## 93 What has been added since the last publication

94 Besides many small changes, mainly three big additions extended nkululeko's functionality  
95 since the last published publications. We introduce them in the next subsections.

### 96 Finetune transformer models

97 With [nkululeko](#) since version 0.85.0 you can finetune a transformer model with [huggingface](#)  
98 (and even [publish it there if you like](#)).

99 If you like to have your model published, set:

```
100 [MODEL]
101 push_to_hub = True
```

102 Finetuning in this context means to train the (pre-trained) transformer layers with your new  
103 training data labels, as opposed to only using the last layer as embeddings.

104 The only thing you need to do is to set your MODEL type to *finetune*:

```
105 [FEATS]
106 type = []
107 [MODEL]
108 type = finetune
```

109 The acoustic features can/should be empty, because the transformer model starts with CNN  
110 layers to model the acoustics frame-wise. The frames are then getting pooled by the model for  
111 the whole utterance (max. duration the first 8 seconds, the rest is ignored).

112 The default base model is the one from [facebook](#), but you can specify a different one like this:

```
113 [MODEL]
114 type = finetune
115 pretrained_model = microsoft/wavlm-base
116
117 duration = 10.5
```

118 The parameter *max\_duration* is also optional (default=8) and means the maximum duration of  
119 your samples / segments (in seconds) that will be used, starting from 0. The rest is disregarded.

120 You can use the usual deep learning parameters:

```
121 [MODEL]
122 learning_rate = .001
123 batch_size = 16
124 device = cuda:3
```

125 `measure = mse`  
 126 `loss = mse`  
 127 but all of them have defaults.  
 128 The loss function is fixed to \* weighted cross entropy for classification \* concordance correlation  
 129 coefficient for regression  
 130 The resulting best model and the huggingface logs (which can be read by [tensorboard](#)) are  
 131 stored in the project folder.

## 132 Ensemble classification

133 With [nkululeko](#) since version 0.88.0 you can combine experiment results and report on the  
 134 outcome, by using the `ensemble` module.

135 For example, you would like to know if the combination of expert features and learned  
 136 embeddings works better than one of those. You could then do

```
137 python -m nkululeko.ensemble \
138 --method max_class \
139 tests/exp_emodb_praat_xgb.ini \
140 tests/exp_emodb_ast_xgb.ini \
141 tests/exp_emodb_wav2vec_xgb.in
```

142 (all in one line) and would then get the results for a majority voting of the three results for  
 143 Praat, AST and Wav2vec2 features.

144 Other methods are *mean*, *max*, *sum*, *max\_class*, *uncertainty\_threshold*, *uncertainty\_weighted*,  
 145 *confidence\_weighted*: \* **majority\_voting**: The modality function for classification: predict the  
 146 category that most classifiers agree on. \* **mean**: For classification: compute the arithmetic  
 147 mean of probabilities from all predictors for each labels, use highest probability to infer the  
 148 label. \* **max**: For classification: use the maximum value of probabilities from all predictors  
 149 for each labels, use highest probability to infer the label. \* **sum**: For classification: use the  
 150 sum of probabilities from all predictors for each labels, use highest probability to infer the  
 151 label. \* **max\_class**: For classification: compare the highest probabilities of all models across  
 152 classes (instead of same class as in `max_ensemble`) and return the highest probability and the  
 153 class \* **uncertainty\_threshold**: For classification: predict the class with the lowest uncertainty  
 154 if lower than a threshold (default to 1.0, meaning no threshold), else calculate the mean of  
 155 uncertainties for all models per class and predict the lowest. \* **uncertainty\_weighted**: For  
 156 classification: weigh each class with the inverse of its uncertainty (1/uncertainty), normalize  
 157 the weights per model, then multiply each class model probability with their normalized weights  
 158 and use the maximum one to infer the label. \* **confidence\_weighted**: Weighted ensemble  
 159 based on confidence (1-uncertainty), normalized for all samples per model. Like before, but  
 160 use confidence (instead of inverse of uncertainty) as weights.

## 161 Predicting Speaker ID

162 To have labels for the individual speakers in a database is extremely important, because if you  
 163 mix the same speakers in training and testing data splits, it is very possible that your model  
 164 simply learned some speaker idiosyncrasies instead of some underlying principle. If you don't  
 165 have this labels, you could at least try to infer them with a pre-trained model.

166 With [nkululeko](#) since version 0.93.0 the [pyannote](#) segmentation package is interfaced (as an  
 167 alternative to [silero](#))

168 There are two modules that you can use for this: \* `SEGMENT` \* `PREDICT`

169 The (huge) difference is, that the `SEGMENT` module looks at each file in the input data and  
 170 looks for speakers per file (can be only one large file), while the `PREDICT` module concatenates

```

171 all input data and looks for different speakers in the whole database.
172 In any case best run it on a GPU, as CPU will be very slow (and there is no progress bar).
173 If you specify the method in [SEGMENT] section and the hf_token (needed for the pyannote
174 model) in the [MODEL] section
175 [SEGMENT]
176 method = pyannote
177 segment_target = _segmented
178 sample_selection = all
179 [MODEL]
180 hf_token = <my hugging face token>
181
182 your resulting segmentations will have predicted speaker id attached.. Be aware that this is
183 really slow on CPU, so best run on GPU and declare so in the [MODEL] section:
184
185 [MODEL]
186 hf_token = <my hugging face token>
187 device=gpu # or cuda:0
188
189 As a result a new plot would appear in the image folder: the distribution of speakers that were
190 found.
191
192 Simply select speaker as the prediction target:
193
194 [PREDICT]
195 targets = ["speaker"]
196
197 Generally, the PREDICT module is described here

```

## Statement of need

Open-source tools are believed to be one of the reasons for accelerated science and technology. They are more secure, easy to customise, and transparent. There are several open-source tools that exist for acoustic, sound, and audio analysis, such as librosa (McFee et al., 2015), TorchAudio (Yang et al., 2021), pyAudioAnalysis (Giannakopoulos, 2015), ESPNET (Watanabe et al., 2018), and SpeechBrain (Ravanelli et al., 2021). However, none of them are specialised in speech analysis with high-level interfaces for novices in the speech processing area.

One exception is Spotlight (Suwelack, 2023), an open-source tool that visualises metadata distributions in audio data. An existing interface between nkululeko and Spotlight can be used to combine the visualisations of Spotlight with the functionalities of Nkululeko.

Nkululeko follows these principles:

- *Minimum programming skills:* The only programming skills required are preparing the data in the correct (CSV) format and running the command line tool. For AUDFORMAT, no preparation is needed.
- *Standardised data format and label:* The data format is based on CSV and AUDFORMAT, which are widely used formats for data exchange. The standard headers are like 'file', 'speaker', 'emotion', 'age', and 'language' and can be customised. Data could be saved anywhere on the computer, but the recipe for the data preparation is advised to be saved in nkululeko/data folder (and/or make a soft link to the original data location).
- *Replicability:* the experiments are specified in a configuration file, which can be shared with others including the splitting of training, development, and test partition. All results are stored in a folder with the same name as the experiment.

- *High-level interface*: the user specifies the experiment in an INI file, which is a simple text file that can be edited with any text editor. The user does not need to write Python code for experiments.
- *Transparency*: as CLI, *nkululeko* *always output debug*, in which info, warning, and error will be obviously displayed in the terminal (and should be easily understood). The results are stored in the experiment folder for further investigations and are represented as images, texts, and even a fully automatically compiled PDF report written in latex.

## Usage in existing research

Nkululeko has been used in several research projects since its first appearance in 2022 (Burkhardt, Wagner, et al., 2022). The following list gives an overview of the research papers that have used Nkululeko:

- (Burkhardt, Eyben, et al., 2022): this paper reported a database development of synthesized speech for basic emotions and its evaluation using the Nkululeko toolkit.
- (Burkhardt et al., 2024): this paper shows how to use Nkululeko for bias detection. The findings on two datasets, UACorpus and Androids, show that some features are correlated with the target label, e.g., depression, and can be used to detect bias in the database.
- (Atmaja et al., 2024): this paper shows Nkululeko's capability for ensemble learning with a focus on uncertainty estimation.
- (Atmaja & Sasou, 2025): in this paper, evaluations of different handcrafted acoustic features and SSL approaches for pathological voice detection tasks were reported, highlighting the ease of using Nkululeko to perform extensive experiments including combinations of different features at different levels (early and late fusions).
- (Atmaja et al., 2025): this paper extends the previous ensemble learning evaluations with performance weighting (using weighted and unweighted accuracies) on five tasks and ten datasets.

## Changes

Nkululeko has been described in three papers so far, we give a short overview on the updates since then.

- **2022 Paper**: F. Burkhardt, Johannes Wagner, Hagen Wierstorf, Florian Eyben and Björn Schuller: Nkululeko: A Tool For Rapid Speaker Characteristics Detection, Proc. Proc. LREC, 2022. **New features**: First version mainly focussing on basic machinelearning experiments that combine *expert* acoustic features (like Praat or opensmile features) with traditional learning approaches.
- **2023 Paper**: F. Burkhardt, Florian Eyben and Björn Schuller: Nkululeko: Machine Learning Experiments on Speaker Characteristics Without Programming, Proc. Interspeech, 2023. **New features**: Mainly extending the acoustic features to deep-learning based (like TRILL, Hubert or wav2vec2) and the models by neural net architectures like MLP or CNN.
- **2024 Paper**: F. Burkhardt, Bagus Tris Atmaja, Anna Derington, Florian Eyben and Björn Schuller: Check Your Audio Data: Nkululeko for Bias Detection, Proc. Oriental COCOSA, 2024. **New features**: Introducing the concept of interfaces (or *modules*), focusing on the *explore-module* that features automatic data statistics and and bias analysis.



- **Since then:** Besides many minor enhancements; ensemble learning, Wav2vec2 model finetuning, adding automatic speaker identification, extending augmentation and segmentation.

## Acknowledgements

We acknowledge support from these various projects:

- European SHIFT (*MetamorphoSis of cultural Heritage Into augmented hypermedia assets For enhanced accessibiliTy and inclusion*) project (Grant Agreement number: 101060660);
- European EASIER (*Intelligent Automatic Sign Language Translation*) project (Grant Agreement number: 101016982);
- Project JPNP20006 commissioned by the New Energy and Industrial Technology Development Organization (NEDO), Japan;
- Project 24K02967 from the Japan Society for the Promotion of Science (JSPS).

We thank audEERING GmbH for partial funding.

## References

- Atmaja, B. T., Burkhardt, F., & Sasou, A. (2025). Performance-weighted Ensemble Learning for Speech Classificatio. *2025 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*.
- Atmaja, B. T., & Sasou, A. (2025). Pathological voice detection from sustained vowels: Handcrafted vs. Self-supervised learning. *2025 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*.
- Atmaja, B. T., Sasou, A., & Burkhardt, F. (2024). Uncertainty-based ensemble learning for speech classification. *2024 27th Conference of the Oriental COCODA International Committee for the Co-Ordination and Standardisation of Speech Databases and Assessment Techniques (o-COCOSDA)*, 1–6. <https://doi.org/10.1109/O-COCOSDA64382.2024.10800111>
- Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 12449–12460). Curran Associates, Inc. <https://proceedings.neurips.cc/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf>
- Burkhardt, F., Atmaja, B. T., Derington, A., & Eyben, F. (2024). Check your audio data: Nkululeko for bias detection. *2024 27th Conference of the Oriental COCODA International Committee for the Co-Ordination and Standardisation of Speech Databases and Assessment Techniques (o-COCOSDA)*, 1–6. <https://doi.org/10.1109/O-COCOSDA64382.2024.10800580>
- Burkhardt, F., Eyben, F., & Schuller, W. (2022). SyntAct : A Synthesized Database of Basic Emotions. In Jonne Sälevä & C. Lignos (Eds.), *Proc. Work. Dataset creat. Low. Lang. Within 13th lang. Resour. Eval. conf.* European Language Resources Association.
- Burkhardt, F., Wagner, J., Wierstorf, H., Eyben, F., & Schuller, B. (2022). Nkululeko: A tool for rapid speaker characteristics detection. *2022 Language Resources and Evaluation Conference, LREC 2022*, 1925–1932. ISBN: 9791095546726
- Chaudhary, A., Chouhan, K. S., Gajrani, J., & Sharma, B. (2020). *Deep learning with PyTorch*. <https://doi.org/10.4018/978-1-7998-3095-5.ch003>



- 300 Eyben, F., Wöllmer, M., & Schuller, B. (2010). openSMILE – the munich versatile and fast  
301 open-source audio feature extractor. *MM'10 - Proceedings of the ACM Multimedia 2010*  
302 *International Conference*, 1459–1462. <https://doi.org/10.1145/1873951.1874246>
- 303 Giannakopoulos, T. (2015). pyAudioAnalysis: An open-source python library for audio signal  
304 analysis. *PLoS One*, 10(12), 1–17. <https://doi.org/10.1371/journal.pone.0144610>
- 305 McFee, B., Raffel, C., Liang, D., Ellis, D., McVicar, M., Battenberg, E., & Nieto, O. (2015).  
306 librosa: Audio and Music Signal Analysis in Python. *Proc. 14th Python Sci. Conf., Scipy*,  
307 18–24. <https://doi.org/10.25080/majora-7b98e3ed-003>
- 308 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M.,  
309 Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D.,  
310 Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python.  
311 *Journal of Machine Learning Research*, 12, 2825–2830.
- 312 Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., Subakan, C.,  
313 Dawalatabad, N., Heba, A., Zhong, J., Chou, J.-C., Yeh, S.-L., Fu, S.-W., Liao, C.-F.,  
314 Rastorgueva, E., Grondin, F., Aris, W., Na, H., Gao, Y., ... Bengio, Y. (2021). *SpeechBrain:*  
315 *A general-purpose speech toolkit*. <https://arxiv.org/abs/2106.04624>
- 316 Suwelack, S. (2023). Spotlight. In *GitHub repository*. [https://github.com/Renumics/](https://github.com/Renumics/spotlight/)  
317 [spotlight/](https://github.com/Renumics/spotlight/); GitHub.
- 318 Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Soplin, N. E. Y.,  
319 Heymann, J., Wiesner, M., Chen, N., Renduchintala, A., & Ochiai, T. (2018). ESPNet:  
320 End-to-end speech processing toolkit. *Proc. Annu. Conf. Int. Speech Commun. As-*  
321 *soc. INTERSPEECH, 2018-Sept*(September), 2207–2211. [https://doi.org/10.21437/](https://doi.org/10.21437/Interspeech.2018-1456)  
322 [Interspeech.2018-1456](https://doi.org/10.21437/Interspeech.2018-1456)
- 323 Yang, S., Chi, P.-H., Chuang, Y.-S., Lai, C.-I. J., Lakhota, K., Lin, Y. Y., Liu, A. T., Shi,  
324 J., Chang, X., Lin, G.-T., Huang, T.-H., Tseng, W.-C., Lee, K., Liu, D.-R., Huang,  
325 Z., Dong, S., Li, S.-W., Watanabe, S., Mohamed, A., & Lee, H. (2021). SUPERB:  
326 Speech Processing Universal PERformance Benchmark. *Interspeech 2021*, 1194–1198.  
327 <https://doi.org/10.21437/Interspeech.2021-1775>