Cough Classifier
CS-433 Machine Learning: Project 2

Devrim Celik, Nina Mainusch, Xavier Oliva i Jürgens
Department of Computer Science, École polytechnique fédérale de Lausanne (EPFL), Switzerland

***Abstract*—Cough sound analysis is commonly performed by doctors to diagnose respiratory conditions (i.e. asthma, pneumonia, or COVID-19). An important property that can be detected from cough sounds is whether mucus is produced ("wet" cough) or not ("dry" cough). The goal of this project is to perform cough classification based on cough sounds. This could enable quick and cheap diagnoses without requiring medical professionals. The Embedded Systems Lab [1] has published the COUGHVID dataset [2], a collection of crowd-sourced cough sounds, 2,200 of which have been annotated by doctors. The lab provided a set of around 70 features computed on each cough recording, which we used to perform automated cough type classification. The final results were computed using a private test set that has been excluded from publishing. Our best classical model, the KNN classifier, achieved an AUC score of 0.61, and the deep learning model an AUC score of 0.63.**

I. Introduction

The novel coronavirus disease (COVID-19) has created the need for mass testing to control its spread. In order to identify newly infected subjects, an inexpensive and effective manner of screening for COVID-19 must be found. Exploiting the fact that one of the prevalent symptoms of this disease is a dry cough, cough sound classification has become an arising field of research. It fuses ML and signal processing methods in order to efficiently distinguish a dry from a wet cough, i.e. a potentially infected from a healthy individual, employing merely a mobile phone with a built-in microphone.

Several researchers have taken up this challenge of diagnosing COVID-19 from cough sounds. For our project, we make use of the largest known COUGHVID crowdsourcing dataset [2], which consists of over 20,000 cough recordings, where more than 2,000 are labeled by expert pulmonologists. Making use of the COUGHVID data we performed binary cough classification (dry vs. wet) using state-of-the-art ML algorithms, and assessed the importance of different features to the classification result. We trained several models using the given features and subject metadata and tested their performance on a non-published testing set provided by the lab.

II. Models and Methods

A. Database description

Although the Embedded Systems Lab maintains a dataset of audio files containing the raw audio signal of coughs, we were additionally provided with three datasets containing precomputed features, as well as annotations done by three medical experts. Thus, we did not have to perform our analysis on the raw audio signal. The three datasets contain the data of the same users, but computed using different segmentation types. The first is the non-segmented data, i.e. there are coughs from 1,659 subjects, where there might be multiple coughs per subject. These data were further divided into a *coarse* and a *fine* dataset. In the coarse dataset, each accumulation of coughs has been segmented into groups of consecutive coughs, yielding 3,440 coughs, whereas in the fine one, each individual cough is its own data point, yielding 7,009 coughs. For each data point there are 67 continuous features concerning their properties in the frequency domain. Additionally, there are four features describing metadata (gender, respiratory condition, symptomatic and age of a subject), where three of them are categorical. Each data point contains the information stating which expert annotated the data and classified the cough as being wet or dry.

B. Data Preprocessing

1) Missing values: The missing values in the dataset were set to the respective mean value of the feature or marked as 0.5 for the metadata.

2) Feature transformation: First of all, we standardized the continuous features by removing the mean and scaling to unit variance. Standardization is a common requirement for many machine learning estimators: they might behave badly if the individual features do not look more or less like a standard normal distribution. Additionally, we dropped correlated features to avoid multicollinearity and dummy coded the categorical variables, that is, we encode the class membership information to ones and zeros. Initially, we intended to do recursive feature elimination, but this did not seem to significantly improve our models. Due to the slight imbalance in our data (approximately 27% data points are labeled as being wet coughs), we also employed oversampling using the SMOTE [3] algorithm for each training dataset within a cross-validation step.

C. Model Architecture

Wet vs. dry cough classification is an extremely difficult task, so we approached it from different angles: Firstly, we have the three nested datasets (no/coarse/fine segmentation). Then for each dataset, we trained a model that uses all available features and one that drops the user data. Lastly,

we trained separate models for the data labeled by each of the three experts. We evaluated all our models based on their ROC-AUC-score and F1-score, but the ROC-AUC-score was the decision criterion to rank the models.

1) Segmentation: Since we are working with physiological data, samples from the same subjects must not end up in the training and testing datasets. Especially in the segmented datasets, multiple rows include coughs from the same subject. Coughs greatly vary between subjects, and a robust cough classification algorithm must be able to generalize. To do that, we use the MultiIndex feature of `pandas` to separate subjects.

2) Training and Testing: In order to assess the quality of our models, we performed 10-fold cross-validation. To make sure that our results are valid and to avoid overfitting, we did not place samples of the coarse and fine dataset from the same subject in both training and testing datasets (leave-n-subjects-out). The models with the highest mean ROC-AUC-score were finally tested using the Embedded Systems Lab's private dataset, to which we did not have access.

3) Classical Models: We started by implementing the most common Machine Learning classification algorithms with the `sklearn` library: Logistic Regression, Support Vector Machines, Linear Discriminant Analysis, k-Nearest Neighbors (KNN), Gaussian Naive Bayes, Decision Tree, Random Forest, and eXtreme Gradient Boosting.

4) Artificial Neural Network: Additionally to these classical approaches, we were interested in how an artificial neural network would perform. For this purpose, we used the popular deep learning library `PyTorch`. We set up a very general model structure that uses ReLU activation functions, a BCE (binary cross entropy) loss function, an Adam optimizer, dropout layers and batch-normalization layers. We used simple linear layers, with a dynamically changeable number of layers (depth) and layer sizes (width).

D. Hyperparameter Tuning

1) Classical Models: In order to tune the hyperparameters for each classical model, we implemented a grid search. The parameters to iterate over were: oversampling for all methods, maximum number of iterations for the iterative methods, number of neighbours for the k-Nearest Neighbors algorithm, kernel and gamma for the Support Vector Classifier, maximum depth for the Decision Tree and Random Forest, number of estimators for the Random Forest and the eXtreme Gradient Boosting algorithm.

2) Artificial Neural Network (ANN): Similar to the classical models, we employed a grid search approach in order to find the optimal hyperparameter values for different setups that can be seen in Table I. We performed a variety of sample tests, leading to the conclusion that networks with a single layer and a dropout rate of 0.5 yield the best results.

For each grid search setup we performed 5-fold cross-validation. We trained each model until there was no further increase in terms of performance on a separate validation dataset (10% of the samples that are designated for validation) over the last 5 epochs (early stopping). Due to time constraints, we only considered a dataset setup for testing in which we used the user metadata and in which we did not split the data by expert.

Hyperparameter	Values
Learning Rate	$0.0005, 0.0001$
weight decay rate	$0.0, 0.2, 0.5$
SMOTE	True, False
number/size hidden layers	$50, 200, 400, 1600$

Table I
DIFFERENT SETUPS FOR THE ANN HYPERPARAMETERS.

III. RESULTS

A. Classical Models

1) Impact of Preprocessing: Oversampling improved the AUC score across all models and methods by 7 to 14%, dummy coding the categorical features improved it by about 0.5%. Standardizing the continuous features did not improve the AUC score in general, but was particularly important since it significantly improved the run time of the different models.

2) Important Features: The provided dataset has approximately 70 features. SHAP values break down a prediction to show the impact of each feature, which can be used as a way to perform feature selection, or to improve the explainability of an existing model. SHAP values interpret the impact of having a certain value for a given feature in comparison to the prediction that would be made if that feature took some baseline value.

We made use of the `BorutaShap` feature selection method, which combines the Boruta feature selection algorithm with SHAP values [4]. We performed the algorithm on the dataset using no segmentation and all available features (including user data). The model used to determine the SHAP values is a Decision Tree. The feature that proved to be the best to predict the cough type was the one stating which doctor labeled the data (cf. Figure 4).

When performing the algorithm on the dataset without metadata, the most important features were `MFCC_std0`, `MFCC_mean0`, `MFCC_std5` and `PSD_225425`.

3) Training Data Results: For all three datasets, the cross-validated mean AUC score over all models was always better by about 5% if we split the data and trained one model for each expert. The following reported AUC scores are thus the mean AUC score over all three expert models weighted by the amount of samples per expert.

In Figure 1, the best results for each segmentation type (no/coarse/fine) and method are displayed. For the non-segmented data, the best model was the Naive Bayes classifier, where we achieved better results (AUC score of 0.6013

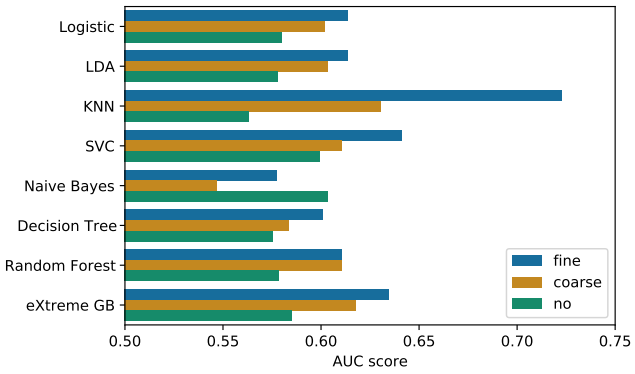

Figure 1. The best AUC scores for each data type (no/coarse/fine segmentation) and classical machine learning methods are displayed here. The results correspond to the mean of the 10-fold cross validation on the training set.

vs. 0.5977) without the metadata features. The other two datasets displayed better results if the metadata features were used in addition and for both the KNN classifier scored highest. For the coarsely segmented data the weighted mean AUC score was 0.6302 (vs. 0.5967 without the metadata). For expert 1, the best setting was to have $k = 2$ neighbours, for expert 2 and 3, $k = 1$ KNN showed the best results. For the fine segmented data, the weighted mean AUC score was 0.7283 (vs. 0.6463 without the metadata) and the optimal $k = 1$.

4) Testing Data Results: Our best models before correcting for potential overfitting for each dataset were tested by the private unpublished test dataset from the Embedded Systems Lab. We got both raw and aggregated results, where aggregated means that the average of the classifier outputs for all coughs of a given subject is taken as the output probability for that subject. Of course there are only aggregated results for the finely and coarsely segmented data samples, since only in those a single cough audio track was split into consecutive or single coughs.

The model for the non-segmented data achieved a raw AUC score of 0.5486, a precision of 0.87 and a F1-score of 0.75. The model for the coarsely segmented data had a raw AUC score of 0.534 and an aggregated AUC score of 0.5489 with a precision of 0.88 and a F1-score of 0.71. The finely segmented model achieved a raw AUC score of 0.5595 and an aggregated AUC score of 0.6085, where the precision was 0.89 and the F1-score 0.67 (cf. Figure 2).

B. Artificial Neural Network

1) Impact of Preprocessing: Using oversampling via the SMOTE algorithm has a positive effect on the cross-validated AUC score, the performance increases by 2%, as can be seen in Figure 5.

2) Important Features: As for the classical models, the SHAP values were calculated for the neural network approach as well. Here, the most important features were `MFCC_mean8`, `MFCC_mean9`, `Spectral_StDev`, `MFCC_std2` and `PSD_225-425`.

	precision	recall	f1-score	support
Dry	0.89	0.55	0.67	99
Wet	0.17	0.56	0.26	16
accuracy			0.55	115
macro avg	0.53	0.55	0.47	115
weighted avg	0.79	0.55	0.62	115

```
Confusion Matrix
[[54 45]
 [ 7  9]]
ROC AUC: 0.60858586
```

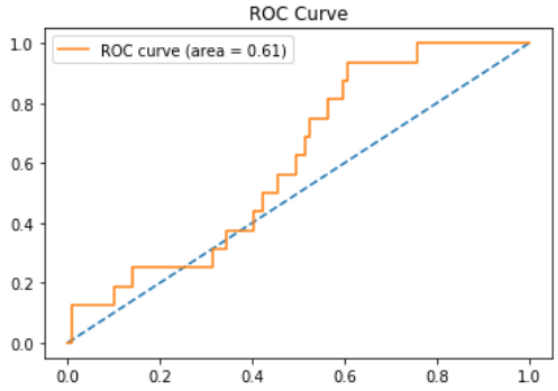

Figure 2. Table of several metrics and the ROC-AUC-curve for the aggregated results of the best classical ML model tested on the finely segmented data.

3) Training Data Results: The AUC performance of the artificial neural network exhibited the following characteristics:

- A learning rate of 0.0005 yielded the best results, over its alternative of 0.0001.
- A weight decay of 0.5 consistently outperforms setups with values of 0.2 and 0.0.
- The average performance of the neural network deteriorates with an increase in the number of neurons in its single hidden layer, with 50 neurons yielding the best average and top performance across all setups (cf. Figure 6).
- Interestingly enough, the artificial neural network performed quite well on data that was not segmented at all with an average performance of a 0.565 AUC score, followed by coarsely segmented data with an average AUC score of 0.56 and finely segmented data, which seems to pose a significant problem to the neural network with an average AUC score of 0.535 (cf. Figure 7).

4) Testing Data Results: After the inferences of the grid search were used in order to equip each model with the best

possible guesses for the right hyperparameters, we finally tested its performance on the unpublished test datasets. The artificial neural networks achieved an aggregated AUC score of 0.63 for the coarsely segmented data, an aggregated AUC score of 0.58 for finely segmented data and a raw AUC score of 0.58 for data with no segmentation.

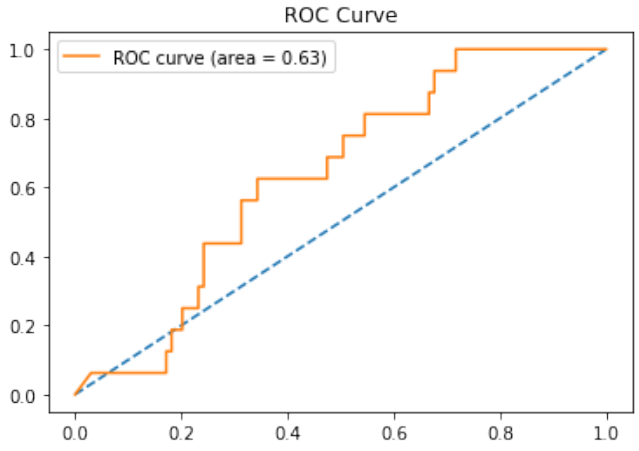

Figure 3. ROC-AUC-curve for the aggregated results of the best deep learning model tested on the coarsely segmented data.

C. Convolutional + Recurrent Neural Network

Finally, we took a final approach to perform cough sound classification on audio files with variable length using a CNN + LSTM architecture. The model architecture and the code is based on work on urban sound classification [5]. The sound file is first converted to the frequency range, then three `Conv2d` layers, an LSTM and a linear layer are sequentially applied. This approach could unfortunately not be appropriately tested, but could be useful for further research.

IV. Discussion and Conclusions

A. The classical results

Regarding the feature importance, the best feature was the one stating which doctor labeled the data. This result is alarming, as it indicates human error in the labeling process and suggests that the opinion of the experts is not reliable. Any label classifier we create will therefore tend to overfit to the three expert labels, thus our results will likely degrade when the data is tested on a testing set labeled by a fourth expert.

Inspecting the results, our AUC scores tend to be higher on the training data than on the test data, as expected. The immense difference that arose between the AUC scores for the finely segmented data (0.7283 train vs. 0.6085 test) however indicates overfitting, which can be compensated by choosing a larger value for k that incorporates the bias-variance trade-off. Overall, the model for the fine segmented data yields the best results, probably due to the reason that it provides more samples.

Embedding our results in the context of finding a good cough classifier we can state that our models are suited to address the task and return moderately good results, but they still rely on metadata features and on the expert labeling, which is not desired in the long run.

B. The ANN results

Unexpectedly, the artificial neural network performed best on the coarsely segmented data (0.63 on coarse segmentation vs. 0.58 on fine and no segmentation), even though more samples would be available on the finely segmented data. Furthermore, we were a bit surprised to see that more layers generally worsened the performance of the artificial neural network. It is our impression, that this is either due to overfitting or the fact that learning is significantly slower with multiple layers, because the gradient is vanishing throughout each layer. At this point, we are very interested in running further setups with more neurons and more layers, from which the time constraint stopped us in this project. There is every reason to believe, that with more time, smaller learning rates and more finely tuned hyperparameters we can outperform the current setups by a large margin.

V. Summary

The goal of this project was to classify a wet and dry cough without using any user metadata, such as age and symptoms. As the task is rather complex and difficult even for experts in the field, we analyzed the results to find out what features are important for the prediction, and tried to develop accurate methods that are robust and generalizable. We developed a deep learning model and compared its performance to that of classical ML methods. We analyzed the feature importance using the Boruta algorithm (with SHAP values). In terms of feature engineering, dummy coding as well as oversampling increased the network's classification accuracy and standardizing improved the runtime.

Our best results on test data using classical ML methods were achieved with a KNN classifier for the finely segmented data, scoring an aggregated AUC value of 0.6085. Generally, all classical approaches performed better on finely segmented data. The deep learning model achieved an aggregated AUC test score of 0.63 on the coarsely segmented data.

For future work, it would be interesting to train and test a recurrent neural network that uses the raw audio signal directly and compare its performance to the other methods. To do that, we started working on a model consisting of convolutional and recurrent layers previously used to perform audio classification.

Finally, a larger dataset with more expert annotators might be necessary to make generalizable and robust predictions. Cough classification is a difficult task, so more experts would potentially be needed to improve the labels to train on.

References

[1] "Embedded systems laboratory - epfl," https://www.epfl.ch/labs/esl/, (Accessed on 12/11/2020).

[2] L. Orlandic, T. Teijeiro, and D. Atienza, "The COUGHVID crowdsourcing dataset: A corpus for the study of large-scale cough analysis algorithms," September 2020, For more information about the data collection, pre-processing, validation, and data structure, please refer to the following publication: https://arxiv.org/abs/2009.11644 The cough pre- processing and feature extraction code is available from the following c4science repository: https://c4science.ch/diffusion/10770/. [Online]. Available: https://doi.org/10.5281/zenodo.4048312

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[4] S. Ekeany, Rory Byrne, "Boruta-shap," https://github.com/Ekeany/Boruta-Shap, 2020.

[5] T. H. Kiran Sanjeevan, "Pytorch audio classification: Urban sounds," https://github.com/ksanjeevan/crnn-audio-classification, 2019.

Appendix

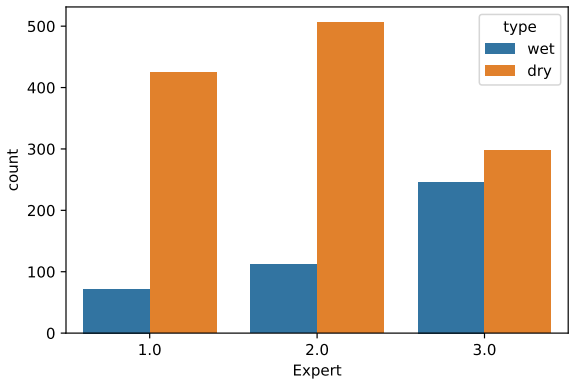

Figure 4. Distribution of the expert cough classification labels. The distribution of the labels greatly varies depending on which expert annotated it.

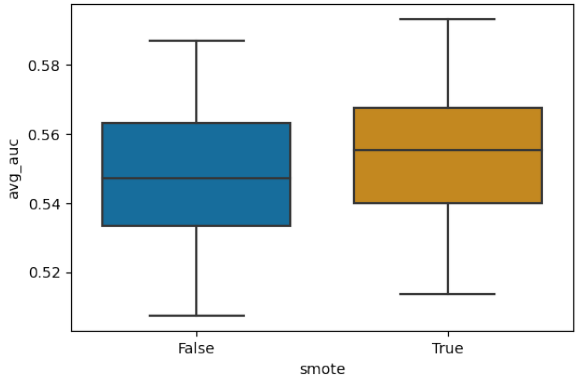

Figure 5. Average AUC performance of grid search runs in respect to whether SMOTE oversampling was utilized or not.

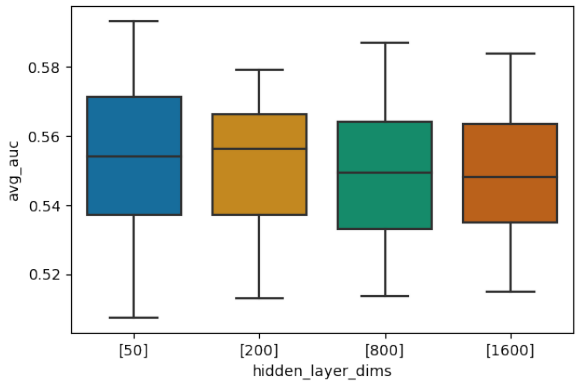

Figure 6. Average AUC performance of grid search runs in respect to the number of neurons in the hidden layer.

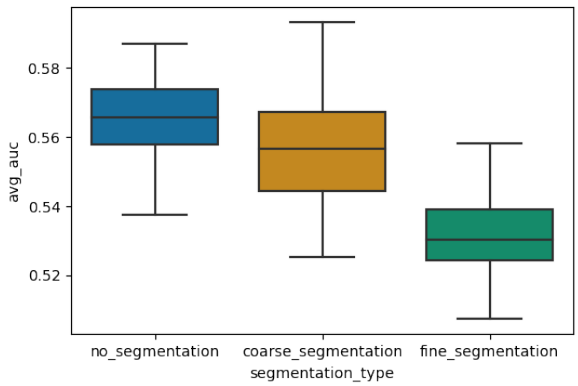

Figure 7. Average AUC performance of the grid search runs in respect to the differently segmented data representations.