The North DiCOVA 2021 Challenge System Report

Isabella Södergren[1], Maryam Pahlavan Nodeh[2], Konstantina Nikolaidou[2], Prakash Chandra Chhipa[2], György Kovács[2]

[1]Digital Services and Systems,Luleå University of Technology, Luleå, Sweden
[2]EISLAB Machine Learning, Luleå University of Technology, Luleå, Sweden

sasde-5@student.ltu.se[1], firstname.middlename.lastname@ltu.se[2]

Abstract

The detection of COVID-19 is, and will foreseeably remain a crucial challenge in the world. This makes the development of potential tools for this task important. One possible approach is on the confines of speech and audio processing, detecting potential COVID-19 infections based on recordings of cough sounds. In our work we set out to propose a simple yet robust method based on well-known features, and classical machine learning methods. For this, we approached the task using the widely-used ComPare 2016 feature set, and classical machine learning models, such as Random Forests, Support Vector Machines (SVMs)). Furthermore, to exploit the complementary nature of different models, we also applied ensembling. Our results showed that our simple approach lead to a robust model, that also produced results on the test set that (besides exceeding the baseline), attains a competitive performance, providing an AUC score of 85.21 on the test set. In the future, our goal is to complement the predictions of these models with more advanced convolutional models from the field of computer vision.

Index Terms: COVID-19, acoustics, machine learning, respiratory diagnosis, healthcare

1. System Description

1.1. Methodology Overview

The ongoing challenge of the COVID-19 pandemic renders the problem of cough classification, and thus challenges in the subject [1] urgent, and topical. For this we have dedicated significant effort to this task. One important lesson we kept in our mind for this from earlier challenges was the importance of combining various models. Another guiding principle was the utility of classical machine learning models, particularly in scenarios where the amount of data available is limited. For this, we have taken the simple approach of utilizing a well-known, easily applicable feature set, the Compare 2016 feature set [2] that can be extracted from audio files using the OpenSmile toolkit [3]. Then we trained classical machine learning models (RandomForests and Support Vector Machines) using the resulting data, using the five-fold cross-validation provided by the organisers of the challenge. We optimized the meta-parameters of these models based on the five validation sets, selecting those that provided the best average Area Under Curve of Receiving Operating Characteristics. Then, for each method we got the final probability predictions by averaging the predictions of the five models trained using the five different folds. Lastly, we combined our two selected models by taking the weighted average of their average predictions, as a further layer of ensembling. As a baseline for comparison, we also trained a multi-layer perceptron on the same features.

1.2. Pre-processing

Z-score normalization Z-score normalization is a method of normalizing data that avoids the outlier issue by controlling the data distribution using mean and standard deviation.

$$(x - \mu)/\sigma \tag{1}$$

x is feature, μ is the mean value, and σ is the standard deviation of the feature. It is a serve as a metric for comparing quantitative features of different scales belonging to different distribution. This characteristic makes it good fit for as normalization technique during feature developments. By applying the z score on features of open smile, It redefined the feature space with balanced representation and regularized the feature values scales thus allow models to learn quick and robust. It also prevented the feature domination during learning which improve model performance generalization with uniform feature dependencies. We applied z-score normalization on the feature set before using the CompParE 2016 features in Support Vector Machines or Multi-layer perceptrons, but not for Random Forests.

1.3. Feature Description

The open-source Speech and Music Interpretation by Large-space Extraction (openSMILE [3]) toolkit is a framework that automatically extracts sound descriptors (low-level features) such as frame energy, voice intensity/loudness, band spectra, loudness approximated from auditory spectra, fundamental frequency, spectral features, psychoacoustic sharpness, spectral harmonicity, and many more. OpenSMILE is a real-time (i.e. able to extract features in real-time) online and offline tool for processing large datasets. Its input can be from different audio formats (including the Free Lossless Audo Codec - flac - format used in the DiCOVA competition), and different operations (e.g. normalization, modification) can be performed in the processing stage. In our work we used the python implementation [4] of this tool to extract the 6373 features of the ComParE 2016 feature set [2] from each audio recording of coughs.

1.4. Classifier Description

For the classification task, we used three different classifiers implemented through scikit-learn python library [5]: Support Vector Machines (SVMs), Random Forests, and Multi-layer Perceptron (MLP). For all classifiers, we trained all folds with different combinations of parameters, and then we select the parameter setting that provided the best AUC score on average on the five validation sets.

Support Vector Machines are commonly used in classification, regression, and outlier detection tasks. SVMs will classify data points by finding a hyperplane in n-dimensional space,

where n is the number of features, that separates the classes. For SVMs, we optimize the regularization parameter C, the kernel type, the degree of the polynomial kernel function when polynomial kernel is chosen, the kernel coefficient gamma, and the multipliers/weights of parameter C for each class.

Multi-layer Perceptron is a class of feed-forward ANNs, capable of learning non-linear separations. We try different values of the following parameters for the MLP classifier: hidden layer size, the activation function for the hidden layer, the weight optimizer, the regularization parameter in the form of penalty alpha, the learning rate initialization and update, the momentum for gradient descent update when used, and Nesterov momentum when Stochastic Gradient Descent with momentum larger than 0 is used.

Random Forests are based on random decision trees and scikit's implementation combines classifiers by averaging their probabilistic prediction, while traditional random forests let each classifier vote for a single class. We optimize the number of trees, the split criterion, the maximum depth of the tree, the use of bootstraps when building trees, the use of out-of-bag samples, and the weights associated with the classes. We present the final optimized parameters values in Table 2.

Table 1: *Meta-parameters used for the different sklearn models applied*

Method	Parameters
RandomForest	bootstrap: False class_weight: None criterion: entropy max_depth: 24 n_estimators: 257 oob_score: False
SVM	C: 2.6499182736174296 class_weight: None degree: 2 gamma: scale kernel: rbf
MLP	activation: tanh alpha: 0.0008091752859707717 hidden_layer_size: 300 learning_rate: constant learning_rate_init: 0.046884249347153434 momentum: 0.2467559334514141 nesterovs_momentum: True solver: sgd

1.5. Results

Results of our experiments (for the validation and test set) are listed in Table 2. As can be seen in Table 2, both the SVM and the Random Forest produced markedly higher scores than the baselines (one provided by the task organizers, and one - MLP - trained by us). We can also see that while our MLP baseline attained higher scores on the validation set, our other methods outperformed it on the test set, which we attribute to their better generalization on the small dataset available. Overall, we can say that our methods seem to show a good generalization ability as well as attain competitive scores on the test set.

Table 2: *ROC AUC scores attained using the various machine learning models on the validation and test set*

Method	ROC AUC score	
	Validation	Test
RandomForest	71.73	82.15
SVM	73.29	85.05
MLP	97.10	75.65
Baseline	68.54	69.85

1.5.1. Ensembling

Lastly, we have combined our RandomForest and SVM models by taking the weighted average of their probability predictions. Here, to avoid overfitting, we examined only five weighting schemes. Results of these experiments on the validation set can be seen in Figure 1. Based on these results, our final combination took the predictions of the RandomForest into account with a weight of 0.25, and consequently, it took the predictions of the SVM model with a weight of 0.75. Using this combination, we managed to marginally improve our results on the test set from 85.05 to 85.21.

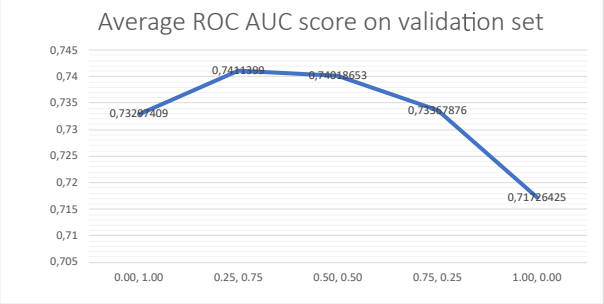

Figure 1: *Weights for the predictions originating from the Random Forest (first number), and SVM (second number) models*

2. References

[1] A. Muguli, L. Pinto, N. R., N. Sharma, P. Krishnan, P. K. Ghosh, R. Kumar, S. Ramoji, S. Bhat, S. R. Chetupalli, S. Ganapathy, and V. Nanda, "DiCOVA challenge: Dataset, task, and baseline system for covid-19diagnosis using acoustics," https://arxiv.org/pdf/2103.09148.pdf, 2021.

[2] B. W. Schuller, S. Steidl, A. Batliner, J. Hirschberg, J. K. Burgoon, A. Baird, A. C. Elkins, Y. Zhang, E. Coutinho, and K. Evanini, "The INTERSPEECH 2016 computational paralinguistics challenge: Deception, sincerity & native language," in *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, N. Morgan, Ed. ISCA, 2016, pp. 2001–2005.

[3] F. Eyben, M. Wöllmer, and B. W. Schuller, "Opensmile: the munich versatile and fast open-source audio feature extractor." in *ACM Multimedia*, A. D. Bimbo, S.-F. Chang, and A. W. M. Smeulders, Eds. ACM, 2010, pp. 1459–1462. [Online]. Available: http://dblp.uni-trier.de/db/conf/mm/mm2010.html#EybenWS10

[4] J. Wagner, C. Hausner, and H. Wierstorf, "Python wrapper for common opensmile feature sets," https://pypi.org/project/opensmile/, 2021.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python ," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.