# COUGHSULTANT

## COVID-19 DETECTION FROM COUGH SOUNDS


CS-577 Project
Ioannis Kaziales ~ csdp1305
03/02/2023

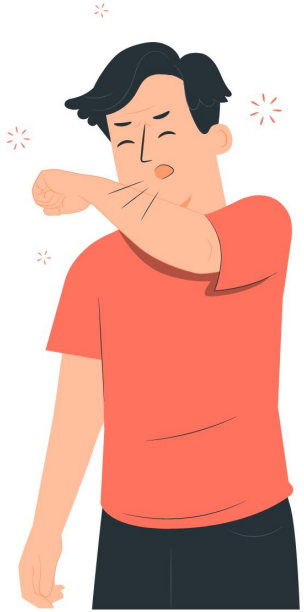
# PROJECT MOTIVATION


## COVID-19

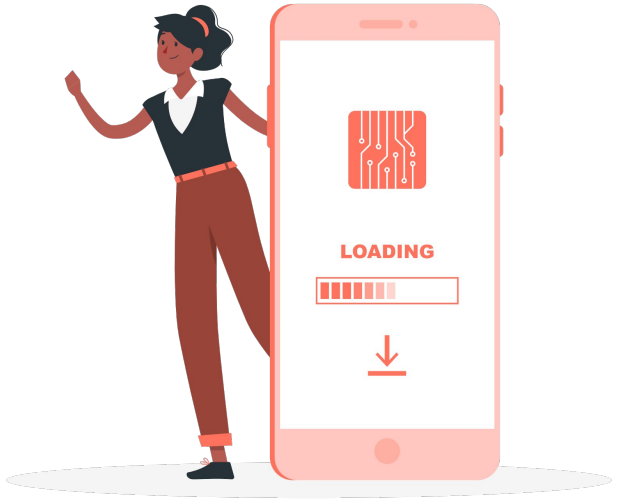
- 753.65M cases & 6.81M deaths (WHO)
- 67.7% of patients exhibit “dry cough” (WHO)
- impact on public healthcare and economy
- early detection is critical to controlling the spread
- traditional diagnostic methods can be expensive, time-consuming, and invasive.
- traditional diagnostic methods may not be readily available (resource-limited settings)
- a machine learning model can provide a cost-effective and non-invasive preliminary test that can be easily scaled for widespread use, even in the absence of laboratory testing.

# HOW COUGHSULTANT WOULD WORK

Enter the platform and fill some data



Record a cough sample




The model will estimate your health status: healthy, symptomatic, or COVID-19

# DISCLAIMER

## COUGHSULTANT

- Is not meant to replace traditional, robust testing methods and diagnostic tools, such as RT-PCR and Rapid tests.

- Is meant to be used just as a fast, inexpensive, scalable and easy-to-use preliminary test accessible to the public.

# 01 OUR DATASET

The “Data Preprocessing” Stage

# BASE DATASET (1/3)

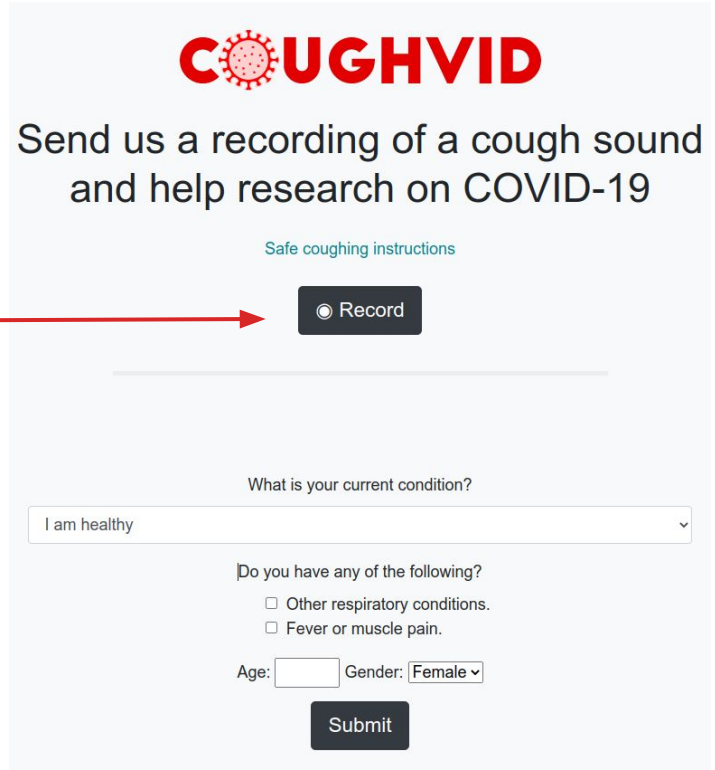

## COUGHVID

- By the Embedded Systems Laboratory (ESL) at EPFL
- Publication in Nature Scientific Data (June 2021)
- Largest known public COVID-19-related cough sound dataset
- 34,4K entries, crowdsourced from all around the world
- Provides useful automatic preprocessing tools
  - cough detection model (XGB classifier with 0.96 AUC)
  - cough segmentation algorithm
- 4 experienced physicians labeled more than 3K recordings to diagnose medical abnormalities present in the coughs

Orlandic, L., Teijeiro, T. and Atienza, D. (2021). The COUGHVID crowdsourcing dataset, a corpus for the study of large-scale cough analysis algorithms. Scientific Data, [online] 8(1), p.156. doi:https://doi.org/10.1038/s41597-021-00937-4.

# BASE DATASET (2/3)

Data was collected through a Web application:




mandatory

optional

# BASE DATASET (3/3)

| Name | Mandatory | Range of possible values | Description |
|---|---|---|---|
| `datetime` | Yes | UTC date and time in ISO 8601 format | Timestamp of the received recording. |
| `cough_detected` | Yes | Floating point in the interval [0, 1] | Probability that the recording contains cough sounds, according to the automatic detection algorithm described in the Methods section. |
| `latitude` | No | Floating point value | Self-reported latitude geolocation coordinate with reduced precision. |
| `longitude` | No | Floating point value | Self-reported longitude geolocation coordinate with reduced precision. |
| `age` | No | Integer value | Self-reported age value. |
| `gender` | No | `{female, male, other}` | Self-reported gender. |
| `respiratory_condition` | No | `{True, False}` | The patient has other respiratory conditions (self-reported). |
| `fever_muscle_pain` | No | `{True, False}` | The patient has fever or muscle pain (self-reported). |
| `status` | No | `{COVID, symptomatic, healthy}` | The patient self-reports that has been diagnosed with COVID-19 (`COVID`), that has symptoms but no diagnosis (`symptomatic`), or that is healthy (`healthy`). |
| `expert_labels_{1,2,3}` | No | JSON dictionary with the manual labels from expert 1, 2 or 3 | The expert annotation variables are described in Table 4. |

← 10 columns per expert

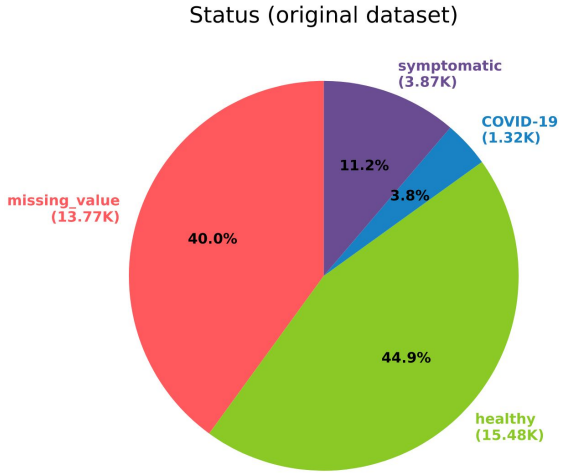
Metadata variables, as they appear in the .csv file

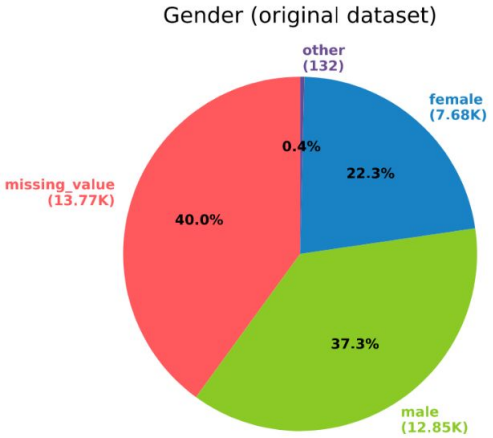
# CLEANING THE DATASET (1/6)

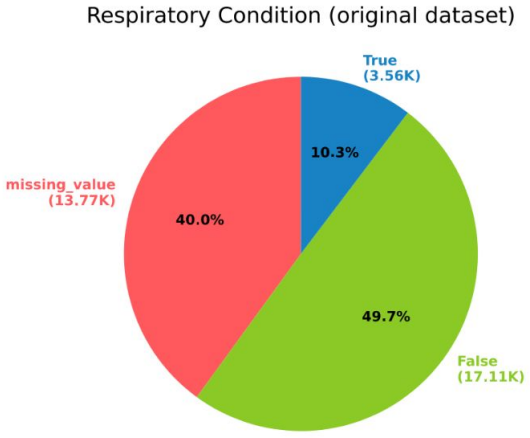
Original COUGHVID dataset:
34.43K entries, 51 variables

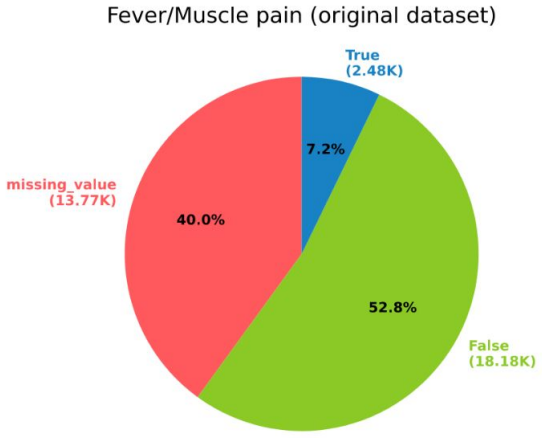
Problem:
Many unlabeled data and missing values

Solution:
Remove those entries entirely



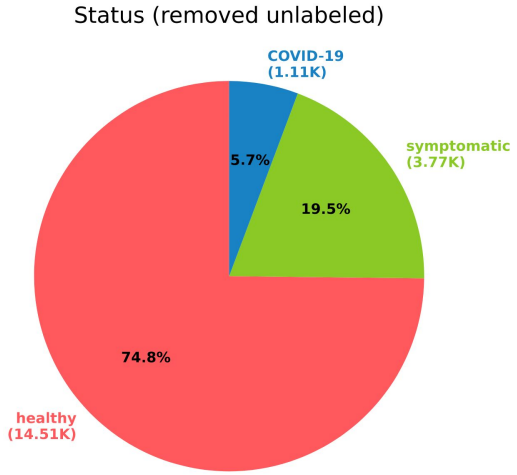



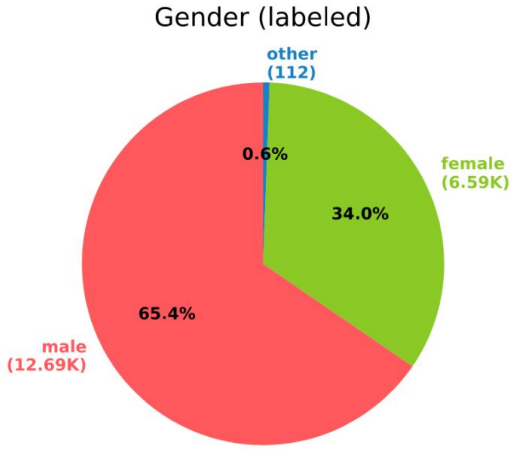



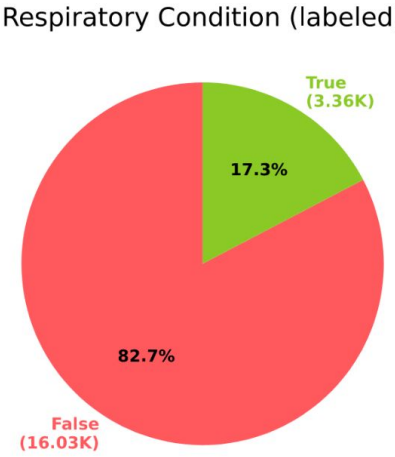



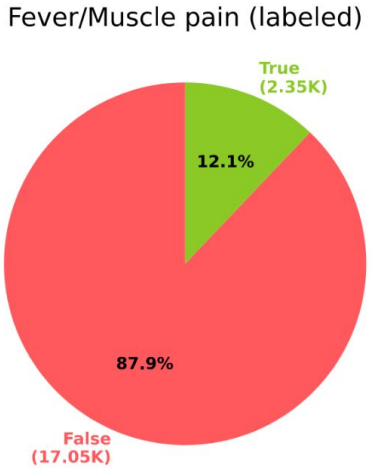
# CLEANING THE DATASET (2/6)

After removing unlabeled data:
19.39K entries, 51 variables



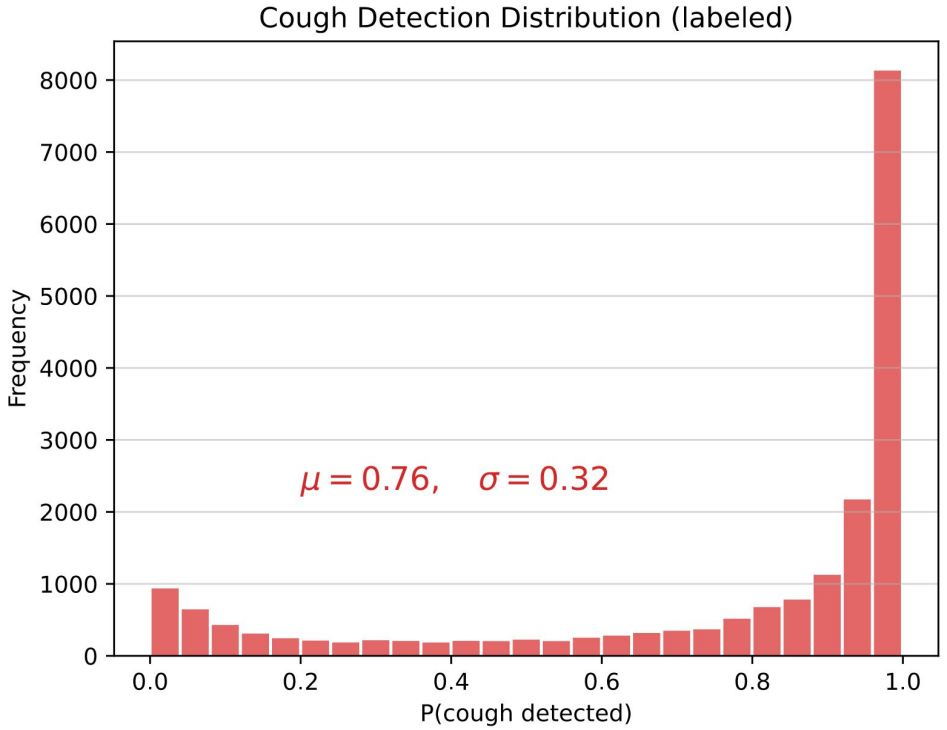
# CLEANING THE DATASET (3/6)

After removing unlabeled data:
19.39K entries, 51 variables

Problem:
Many entries contain recordings that:

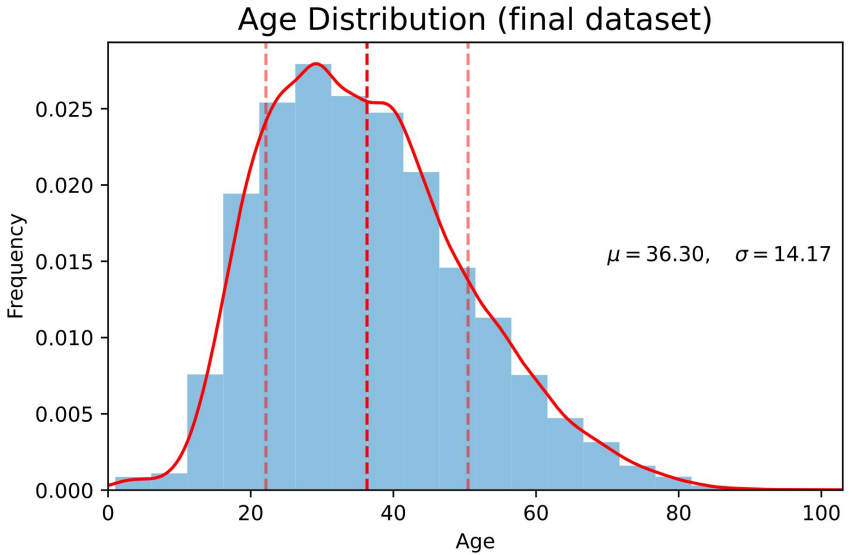
- contain no cough sounds
- have poor quality

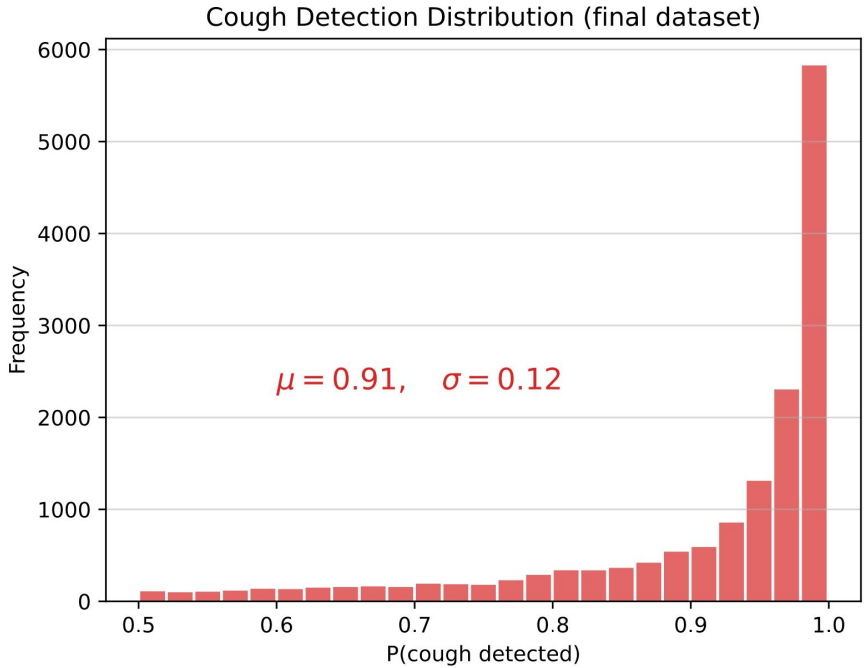
Solution:
Remove those entries entirely.
In our case, we used threshold=0.5



# CLEANING THE DATASET (4/6)

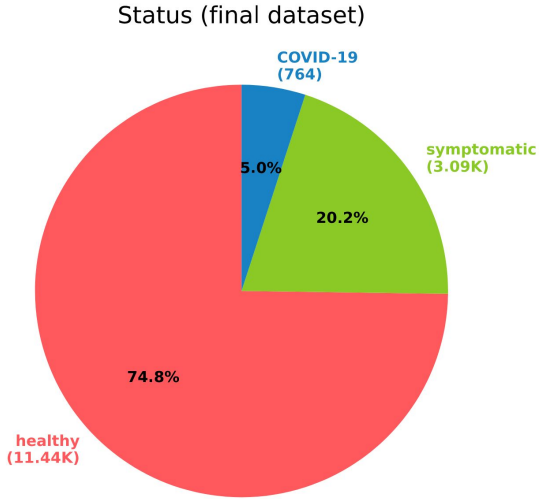
After removing samples with low P(cough):
15.29K entries, 51 variables

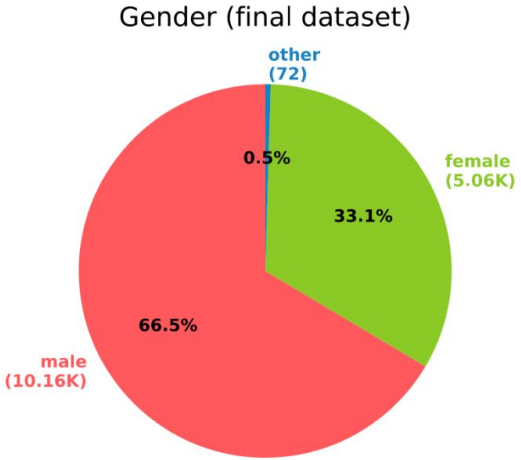


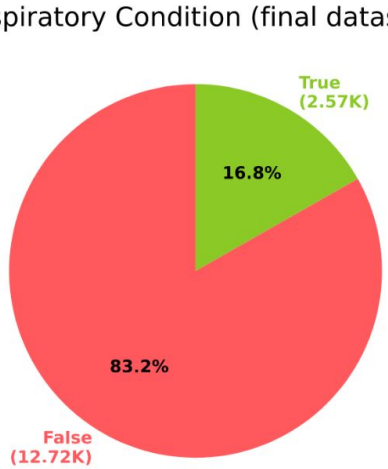



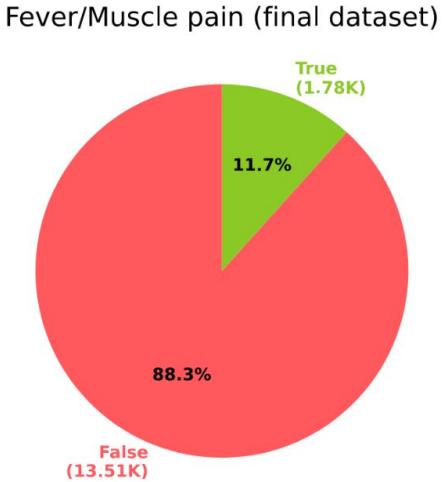
# CLEANING THE DATASET (5/6)

After removing samples with low P(cough):
15.29K entries, 51 variables
(unbalanced dataset)



# CLEANING THE DATASET (6/6)

Fleiss' Kappa scores: consistency of categorical labels of multiple reviewers.
$K_{Fleiss}$ = -1 : no agreement among any reviewers
$K_{Fleiss}$ = 1 : perfect agreement of all reviewers

Problem:
Expert doctors disagree with each other.
Many columns still have missing data and are not useful (e.g. location, datetime, doctors' annotations)

After removing those variables:
15.29K entries, 5 variables

- age
- gender {*female*, *male*, *other*}
- respiratory_condition {*True*, *False*}
- fever_muscle_pain {*True*, *False*}
- status {*healthy*, *symptomatic*, *COVID-19*}

| Item | $K_{Fleiss}$ | Agreement[29] |
|---|---|---|
| quality | −0.06 | Poor |
| cough_type | 0.26 | Fair |
| dyspnea | −0.02 | Poor |
| wheezing | 0.06 | Slight |
| stridor | −0.01 | Poor |
| choking | −0.01 | Poor |
| congestion | 0.41 | Moderate |
| nothing | 0.13 | Slight |
| diagnosis | 0.07 | Slight |
| severity | 0.15 | Slight |

Inter-Expert Label Consistency

# AUGMENTING THE DATASET (1/3)

From each recording, extract acoustic features, using:

- openSMILE feature extraction toolkit (popular in speech/music processing community)
- eGeMAPSv02 feature set (88 features)

eGeMAPSv02:

- high-level descriptors
- useful features for characterizing sounds associated with respiratory diseases (coughs, wheezes, crackles)
- some features: pitch, loudness, jitter, shimmer, HNR, MFCCs, spectral centroids
- designed for tasks such as sentiment analysis and speaker characterization

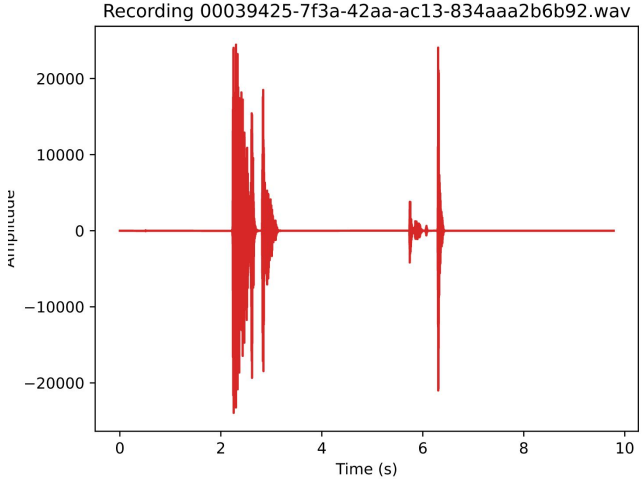

After adding those variables:
15.29K entries, 93 variables

Florian Eyben, Martin Wöllmer, Björn Schuller: "openSMILE - The Munich Versatile and Fast Open-Source Audio Feature Extractor", Proc. ACM Multimedia (MM), ACM, Florence, Italy, ISBN 978-1-60558-933-6, pp. 1459-1462, 25.-29.10.2010.

# AUGMENTING THE DATASET (2/3)

**Process followed to extract the features from each recording:**

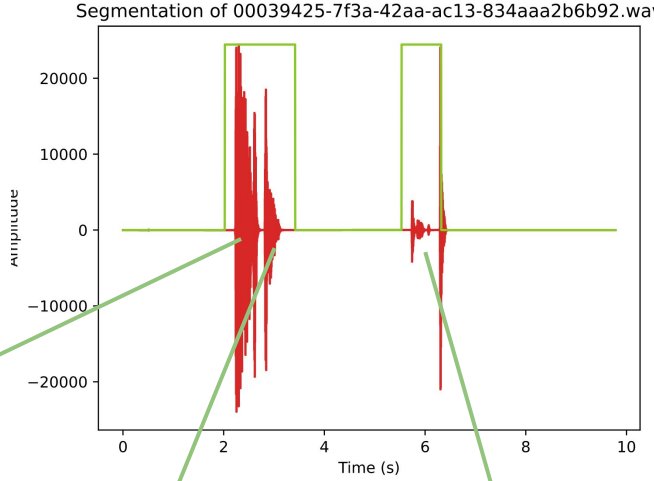
1. Load the audio recording
2. Identify and segment the separate cough sounds present in the recording
3. Calculate P(segment i contains cough)
4. Extract features from the segment: $j = \text{argmax}_i$ P(segment i contains cough)

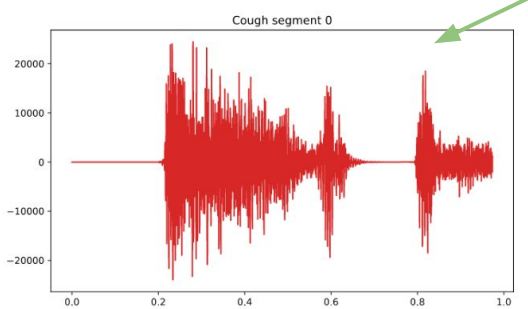


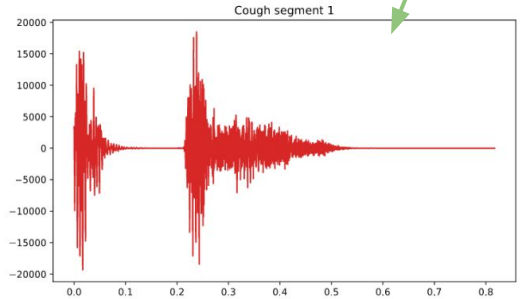
# AUGMENTING THE DATASET (2/3)

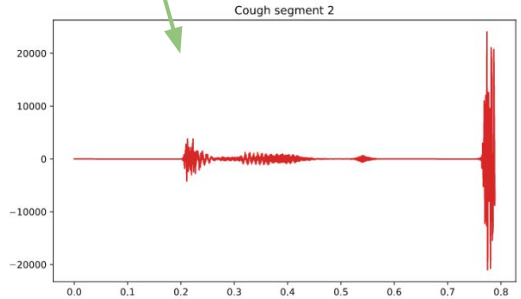
**Process followed to extract the features from each recording:**

1. Load the audio recording
2. Identify and segment the separate cough sounds present in the recording
3. Calculate P(segment i contains cough)
4. Extract features from the segment: $j = \text{argmax}_i P(\text{segment i contains cough})$



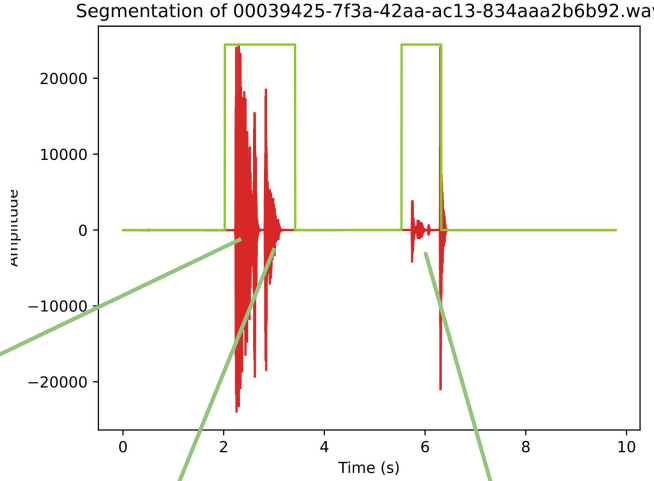



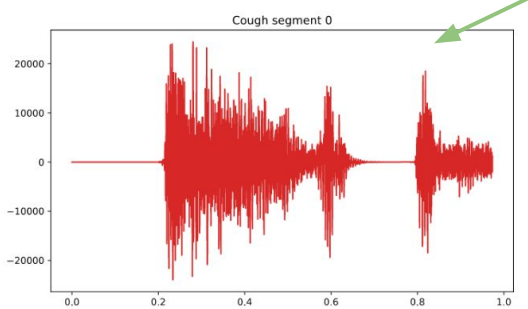



# AUGMENTING THE DATASET (2/3)

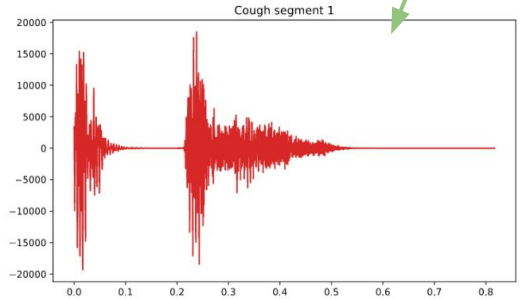
**Process followed to extract the features from each recording:**

1. Load the audio recording
2. Identify and segment the separate cough sounds present in the recording
3. Calculate P(segment i contains cough)
4. Extract features from the segment: $j = \text{argmax}_i$ P(segment i contains cough)

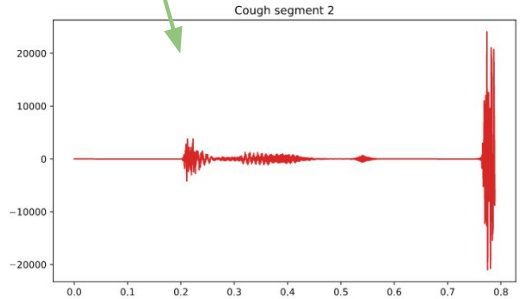



P(cough) = 89.68 %




P(cough) = 74.48 %

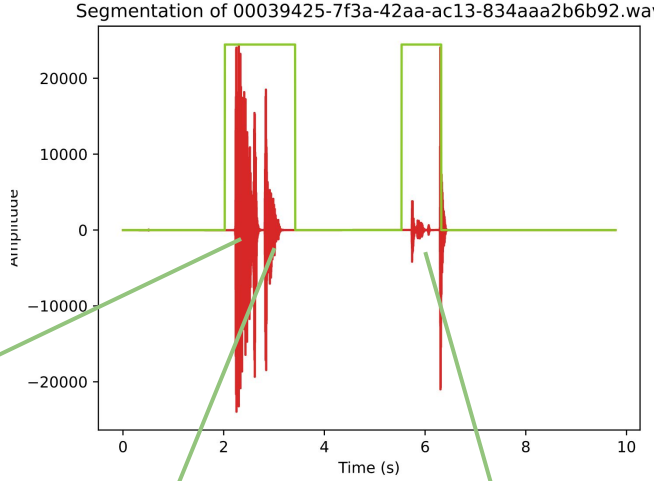


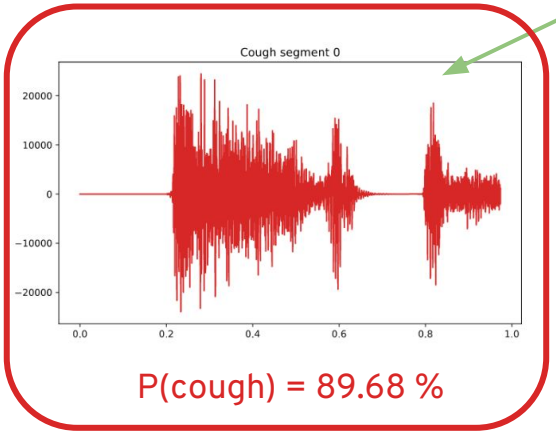

P(cough) = 46.59 %

# AUGMENTING THE DATASET (2/3)

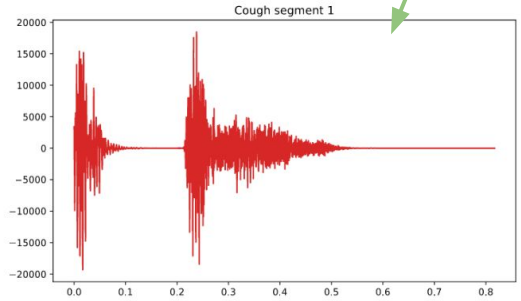
**Process followed to extract the features from each recording:**

1. Load the audio recording
2. Identify and segment the separate cough sounds present in the recording
3. Calculate P(segment i contains cough)
4. Extract features from the segment:
   $j = \text{argmax}_i$ P(segment i contains cough)

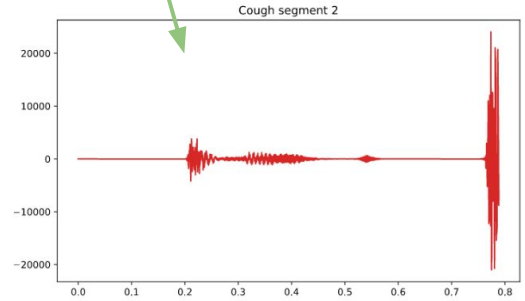



P(cough) = 89.68 %




P(cough) = 74.48 %




P(cough) = 46.59 %

# AUGMENTING THE DATASET (3/3)

## Encoding the categorical variables

### gender

{*female*, *male*, *other*}

nominal feature (not ordered)

→ one hot encoding

*gender_female*, *gender_male*, *gender_other*

### status

{*COVID-19*, *healthy*, *symptomatic*}

target variable

→ label encoding

{*0*, *1*, *2*}

| | gender |
|---|---|
| 0 | male |
| 1 | male |
| 2 | male |
| 3 | male |
| 4 | female |
| ... | ... |
| 15287 | male |
| 15288 | male |
| 15289 | female |
| 15290 | male |
| 15291 | female |

15292 rows × 93 columns

| | gender_female | gender_male | gender_other |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 0 |
| 4 | 1 | 0 | 0 |
| ... | ... | ... | ... |
| 15287 | 0 | 1 | 0 |
| 15288 | 0 | 1 | 0 |
| 15289 | 1 | 0 | 0 |
| 15290 | 0 | 1 | 0 |
| 15291 | 1 | 0 | 0 |

15292 rows × 95 columns

# AUGMENTING THE DATASET (3/3)

## Encoding the categorical variables

**gender**

{*female*, *male*, *other*}

nominal feature (not ordered)

→ one hot encoding

*gender_female*, *gender_male*, *gender_other*

**status**

{*COVID-19*, *healthy*, *symptomatic*}

target variable

→ label encoding

{*0*, *1*, *2*}

| status |
|---|
| healthy |
| healthy |
| healthy |
| healthy |
| healthy |
| ... |
| healthy |
| healthy |
| healthy |
| healthy |
| healthy |

| status |
|---|
| 1 |
| 1 |
| 1 |
| 1 |
| 1 |
| ... |
| 1 |
| 1 |
| 1 |
| 1 |
| 1 |

# 02 MODEL FITTING

The “Model Training” Stage

# OVERALL PIPELINE

**Task:** multiclass Classification

**Followed pipeline:**

1. Load the dataset (15.2K rows)

2. Split it with stratification into:
   a. train set (85% - 13K rows)
   b. hold-out set (15% - 2.3K rows)

3. Perform feature selection on the train set

4. Use stratified cross validation to identify the best model & hyperparameters

5. Train the model on the train set and estimate its performance on the hold-out set

6. Train the final model on all the available data

# FEATURE SELECTION

To select the most important features, Sequential (forward) Feature Selection from the MLxtend library was used on the train set, with parameters:

- model : K-Neighbors Classifier with 5 neighbors
- number of stratified folds: 5
- scoring function: weighted One-vs-Rest ROC-AUC

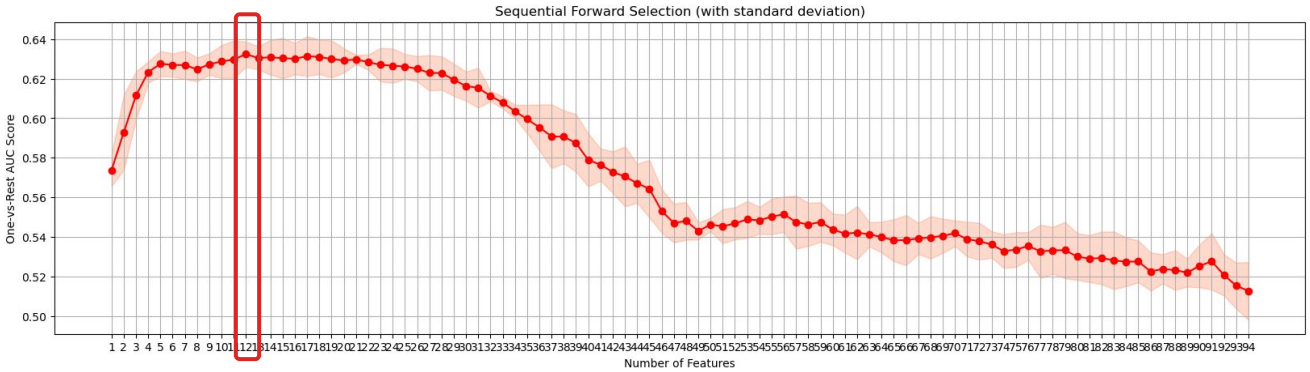
```
function forwardSelection(model, n):
      start with an empty feature set
      until #selected_features < n:
            add one feature based on the
            model's performance

function forwardSelect-CV(model, folds):
      run and cross-validate forwardSelection
      and find the best scoring n and feature
      subset
```

**RESULTS:**

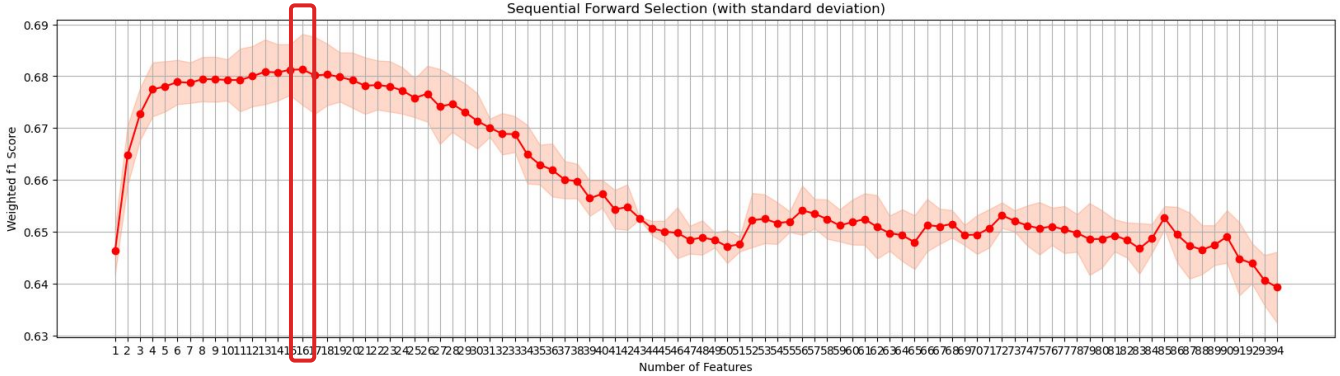
- 12 / 94 original features were selected [metric: weighted ovr ROC-AUC]
- 16 / 94 original features were selected [metric: weighted F1 score]

Raschka, (2018). MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. Journal of Open Source Software, 3(24), 638, https://doi.org/10.21105/joss.00638

# AUC OF SUBSETS DURING FEATURE SELECTION




weighted ovr ROC-AUC
selected 12 features




weighted F1 score
selected 16 features

# HYPER-PARAMETER TUNING

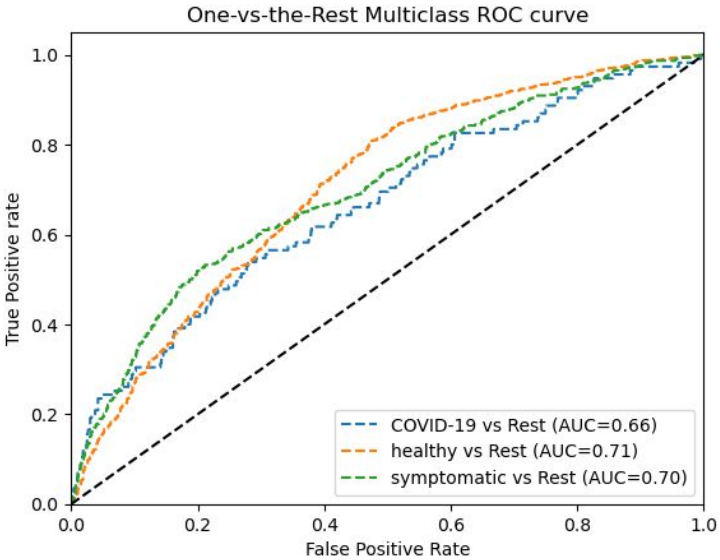
To find the best model and tune the hyperparameters, I used Grid Search with Cross Validation (exhaustive search over specified models and parameter values) on the train set on the selected features.
Parameters used:

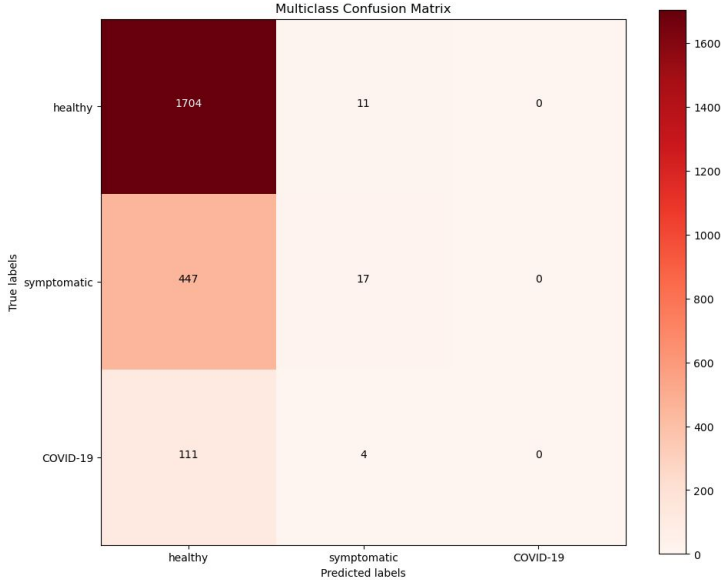
- number of stratified folds: **3**
- scoring function: weighted One-vs-Rest ROC-AUC
- estimators & hyper-parameters
  - Random Forest Classifier:
    trees: {*10*, *50*, *100*, *250*}, max_depth: {*5*, *10*, *20*}, class_weight: {*equal_weight*, *balanced*}
  - Support Vector Classifier:
    C: {*0.01*, *0.1*, *1*, *10*, *100*}, kernel: {*linear*, *poly*, *rbf*}, class_weight: {*equal_weight*, *balanced*}
  - K-Neighbors Classifier:
    n_neighbors: {*3*, *5*, *7*, *10*, *20*}, weights: {*uniform*, *distance*}
  - Gradient Boosting Classifier:
    boosting stages: {*10*, *50*, *100*, *250*}, max tree depth: {*5*, *10*, *20*}

**Best Performing Configurations:**
metric=weighted ovr ROC-AUC: Random Forest Classifier (trees=100, max_depth=5)
metric=weighted F1: Random Forest Classifier (trees=250, max_depth=10, class_weight=balanced)

# PERFORMANCE ESTIMATION (OVR ROC-AUC)

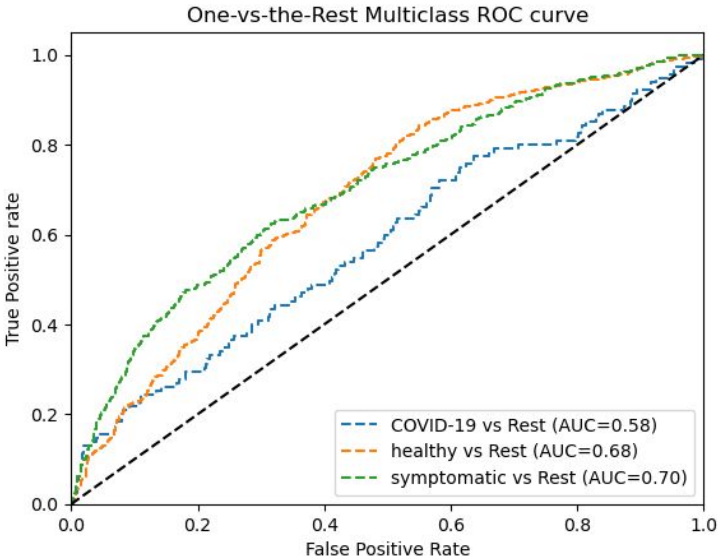
Estimated performance (by training the best configuration on train set and testing on hold-out):
➝ Mean weighted One-vs-Rest ROC-AUC: 0.703 (conservative estimate)
➝ it predicts 'healthy' too often (due to imbalance)

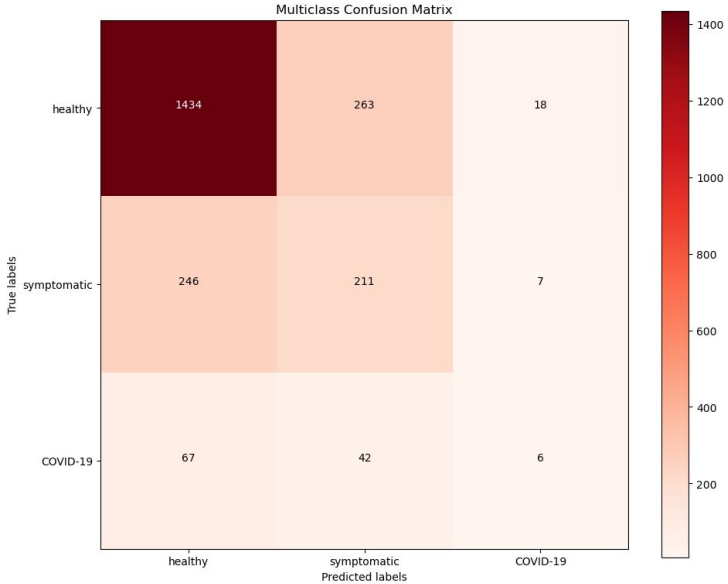


# PERFORMANCE ESTIMATION (F1 SCORE)

Estimated performance (by training the best configuration (with F1 score) on train set and testing on hold-out):
→ Mean weighted One-vs-Rest ROC-AUC: 0.681 (conservative estimate)
→ has better predictions



# CONCLUSION

COUGH**SULTANT** has achieved adequate AUC performance on the test set, but the confusion matrix produces bad results, so perhaps the model shouldn't be applied for clinical use.

## Future Work

- perform a more thorough analysis of the final dataset
- perform data augmentation to improve the class imbalance
- Increase robustness and performance of the model
- use neural networks instead of machine learning approaches
- Deploy the model on a platform

# THANK YOU

Do you have any questions?

csdp1305@csd.uoc.gr
johnnykaz.github.io

CREDITS: This presentation included
People Illustrations by Storyset

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon,** and infographics & images by **Freepik.**