Early Detection of COVID-19 from Cough Sounds, Symptoms, and Context
Machine Learning / Signal Processing Sub-Team
CS 472: Data science and AI for COVID-19

Bernard Lange[1], Derrick Li[2], Effie Nehoran[3], Elena Tuzhilina[4], Mandy Lu[2]
Stanford University Department of Aeronautics and Astronautics[1], Stanford University Department of Computer Science[2], Stanford University Department of Biomedical Data Science[3], Stanford University Department of Statistics[4]

1. Introduction

As of June 12th 2020, there have been 7,739,944 cases and 428,337 deaths as a result of the *COVID-19 pandemic* as per the *World Health Organization COVID-19 Dashboard* [6]. The countries around the world have imposed restrictions in the form of the lockdown to combat the rate of the spread and *flatten the curve* describing the number of patients admitted to hospital per day.

The current status of the pandemic varies between countries and is typically described as before the peak or after the peak. The general opinion is that we should expect a second wave of the outbreak in the oncoming months before vaccination becomes available to the general public. All governments are planning their post-lockdown reopening procedures and methods to monitor the spread of the disease. However, limited tested capabilities, varying quality of the tests, and a large number of asymptomatic carriers make this task particularly challenging.

As a part of this project, we would like to attempt to diagnose the patient based on the cough recording and general patient information collected by the hospital staff. The hope is that if the diagnosis method proves to be effective, a preliminary test could be performed without visiting the hospital and hence, reducing the strain on the medical staff and infrastructure. This task was approached by Imran et al. [12] and provided hope that the COVID-19 test could be effectively framed as a classification model learned from the data. In this work, we are further investigating if this kind of test is a possibility based on the data provided by the *Wadhwani Institute of Artificial Intelligence.*

In this paper, we will present a roadmap of our process and preliminary results. Since we have not yet finished a thorough or rigorous treatment of this task, we will attempt to document our process in a productive manner for further research and experimentation. This work is by no means comprehensive and will hopefully serve as a point of reference for ongoing discussion and research.

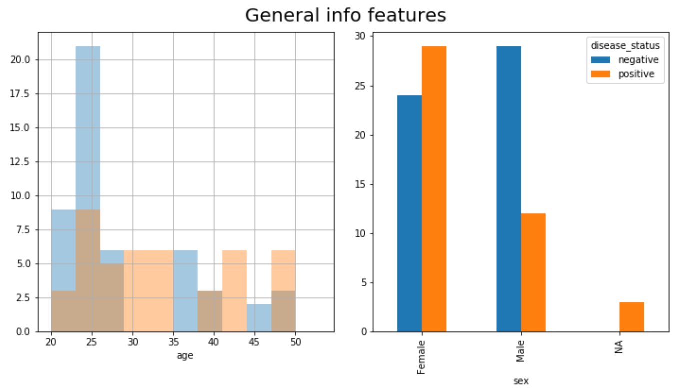
The contributions of this work are two-fold:

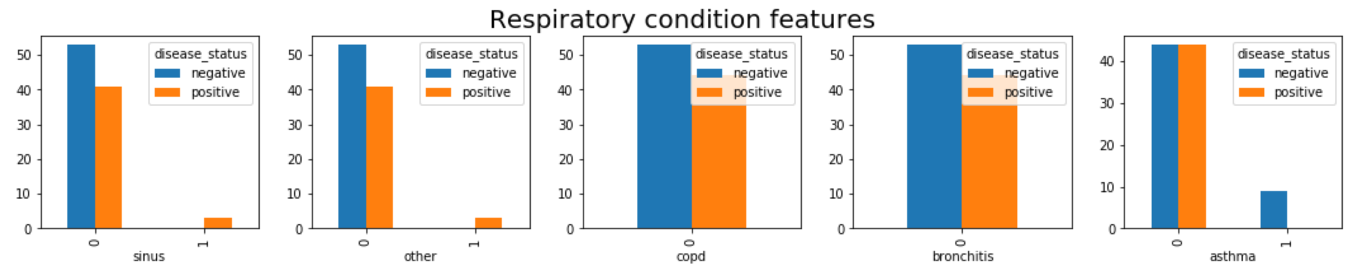
1) A framework that takes a raw cough recording, cuts the audio file to the relevant parts with the cough, reduces the noise and performs classification.
2) Comparison of different classification approaches.

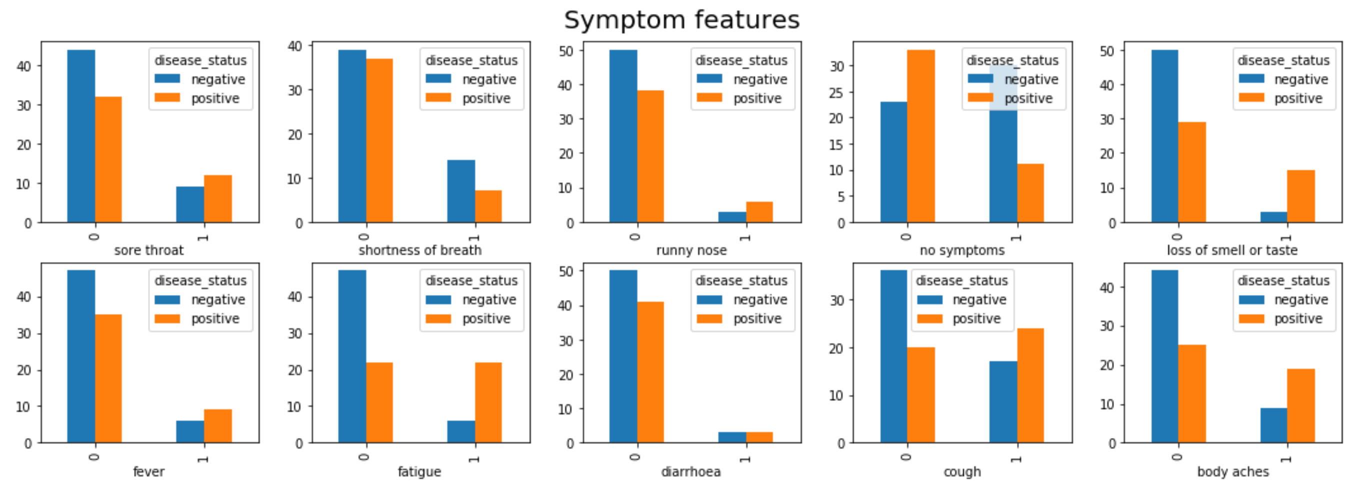
2. Dataset

Data description

The data contains 34 patients tested for COVID-19, 16 positive cases and 18 negative cases. For each patient the avaliable information can be grouped into four feature categories: general info (age and sex), symptoms (sore throat, shortness of breath, runny nose, loss of smell or taste, fever, fatigue, diarrhoea, cough, body aches, no symptoms), respiratory conditions (sinus, copd, bronchitis, asthma and other), sound features (3 cough sounds per each patient). In total the data contains 97 cough sounds (some patients did not provide all three cough recordings). We plot samples feature distributions vs. disease status. Note that people with present respiratory conditions are strongly underrepresented.







Sound data preprocessing

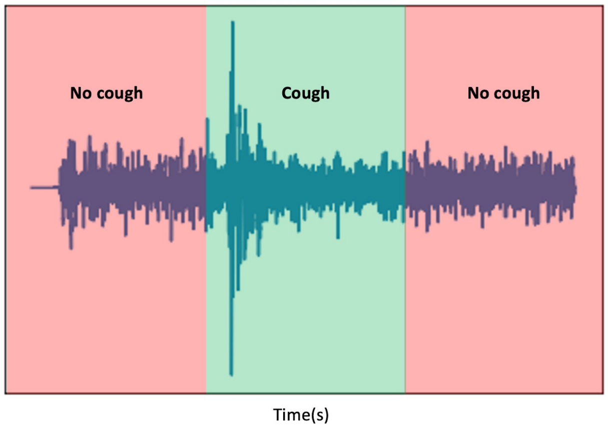
Our cough classification framework consists of the subsequent parts discussed in the following subsections:

1) *Input audio file*: audio file with the following properties: wav format, single channel, 16 kHz sample rate and 256 kbps bitrate
2) *Cough cropping*: detects which parts of the provided audio file contain cough. Subsequently, it cuts the file into two distinct audio files, with cough and with other sounds which are assumed to be noise,

3) *Noise reduction*: uses the provided audio file with noise to suppress the noise in the cough audio file,
4) *Signal Processing*: converts the audio file into the set of features used for classification,
5) *Classification model*: classifies the provided features as a *Covid-19 Cough* or *Non-Covid-19 Cough*.

Cough cropping

Given the limited size of the dataset at hand, there is a risk that the model may overfit to unwanted signals, such as the method of recording or other hospital ward-related sounds. Hence, prior to the cough-specific audio analysis and classification, it is necessary to identify if and when the cough is present in the recording and crop the audio file accordingly. To achieve this, we are using a general audio classification architecture, called *Ubicoustics* [6], capable of classifying a wide range of frequently occurring sounds in the environment, such as leaves rustling, engine noises, water running, dialogues, coughs, and many more. The model is based on the pre-trained *YouTube-8M VGG-16* [7] architecture with a modified last layer, and trained on sound set with substantial audio augmentation including amplification augmentation, persistence augmentation and mixing augmentation using sound effect libraries such as AudioSet [8] and Freesound [9]. The model enables us to classify the audio with 1 Hz prediction frequency which is utilized to determine whether the cough is present in the recording and identify the time window of its occurrence. This information allows us to crop and cut the recording to cough audio file and noise audio file using SciPy library [10] as shown in the Figure on the right.



Noise reduction

Due to the wide range of collected samples that are from differing environments, it was important to control as many of the potential variables as possible. A major problem comes from this type of crowdsourced data is the lack of ability to control for the sounds in the environment and the quality of the microphone. One major factor that was important to standardizing the dataset was to remove as much background noise as possible from all the samples.

At this point, the cough file is already cropped around the actual cough and saved as a .wav file. The cough cropping algorithm from the previous stage also saves a portion of the cough clip that is prior to the actual cough sound. This prior is critical for noise reduction. We consulted and used a sound-processing library called Noise Reduce (see [4]). The library works by removing a certain frequency from the target sound clip by isolating the signal using Fast Fourier Transforms. The background noise can thus be removed by using the prior sample saved from the cough cropping. As a result, using the library with the prior and actual cough as inputs leads to an output that has had its background noise reduced.

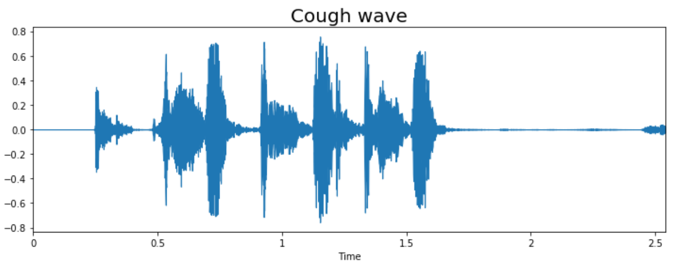
Overall, the noise reduction plays an important role in standardizing the dataset, especially when the data is crowdsourced and difficult to control. Now, the data is one step closer to being cleansed enough to be used for very sensitive analysis.

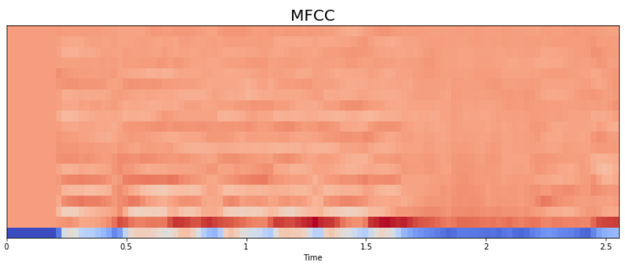
Signal Processing

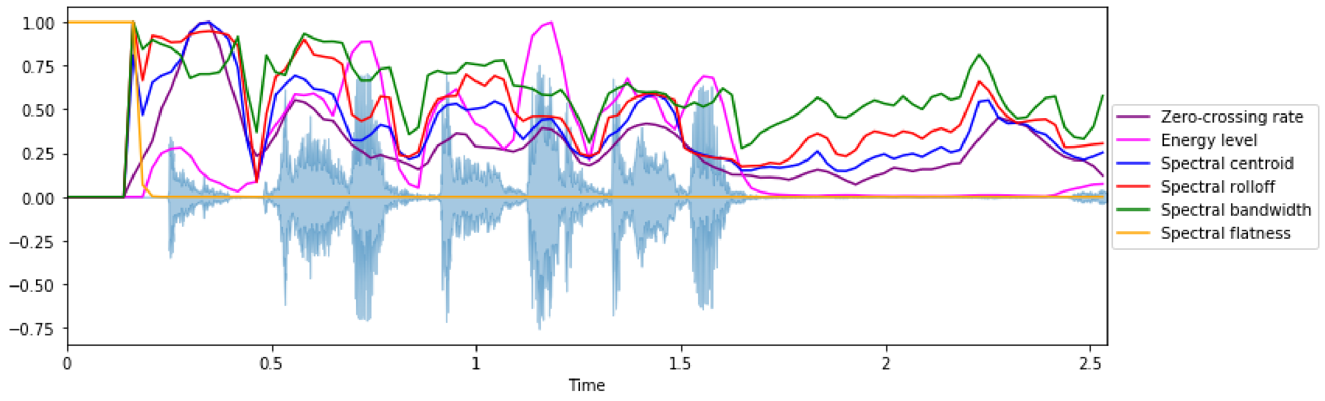
In this part we convert already preprocessed coughs recordings into a set of features. For each cough we extract the following list of the features (26 features in total):

- Mel-frequency cepstral coefficients (MFCCs) - 20 frequencies that characterizes the overall shape of a spectral envelope,
- Zero-crossing rate - measures the rate of sign-changes along a signal,
- Energy level - measures the signal power,
- Spectral centroid indicates at which frequency - characterizes the location of sound “centre of mass” and computes as a weighted mean of the frequencies,
- Spectral rolloff - measures the “shape” of the signal and represents the frequency at which high frequencies decline to zero,
- Spectral bandwidth - determines the resolution of the signal,
- Spectral flatness - quantifies the tonal quality.

This list was created based on the careful literature review on cough classification and cough detection techniques (see [1]-[4]) and the features were obtained by means of the Python *librosa* library. Note that each feature is a time series of length compatible with the length of the original cough wave, so the final feature is an average of the corresponding time series, which helps us not only significantly reduce dimensions, but also avoid the problems with the different feature lengths.







Feature Preprocessing

We output the features from the signal processing stage into a feature matrix, with each training example corresponding to the features of a single cough. We merged this feature matrix with a matrix that includes information about each patient, such as age, sex, and symptoms. Each patient had up to three cough samples, and thus up to three examples in the data.

We prepared the feature matrix for the models by removing irrelevant features such as submission date, device, test location, and file names. Then, we separated “symptoms” and “respiratory conditions” into indicator variables for each type of symptom or condition. After this step, we had the following list of features: 'user_id', 'age', 'sex', 'sore throat', 'shortness of breath', 'runny nose', 'no symptoms', 'loss of smell or taste', 'fever', 'fatigue', 'diarrhoea', 'cough', 'body aches', 'sinus', 'other', 'copd', 'bronchitis', 'asthma', 'disease_status', 'mfcc[1-20]', 'zero_crossing_rate', 'rms', 'spectral_centroid', 'spectral_rolloff', 'spectral_bandwidths', and 'spectral_flatness'.

The original data included a number of patients with inconclusive COVID status, so the next step was to filter out the samples that were not labeled as either positive or negative. Additionally, we removed one patient with no data about gender. Finally, we separated the samples into a training set and a held-out test set, ensuring that both sets had an equivalent proportion of positive versus negative patients, and that samples from the same patient went into the same data set.

After the feature preprocessing we ended up with

- 94 samples divided into 74 train samples and 20 test samples;
- 43 features containing 2 general info features, 5 respiratory condition features, 10 symptom features and 26 sound features.

3. Models

We tested three different types of classification models on the COVID cough data set: logistic regression, support vector machines, and neural networks.

Logistic regression

We start with the logistic regression model. Since the number of features is compatible with the train/test size, to prevent overfitting, the model requires some regularization. To make the features comparable we need to standardize non-categorical features (which are the sound features only). We consider three types of penalties: lasso (or l1 penalty), ridge (or l2 penalty) and elastic net penalty (the sum of l1 and l2 penalty with equal weights). To choose the optimal penalty type as well as the optimal value of penalty factor *C* we use 5-fold cross-validation measuring AUC score (one can consider accuracy as well, but is known to be biased on size of the test data). To compare the importance of the features across the feature types we also check the model performance separately for each general info, respiratory condition, symptom and sound features.

Support Vector Machine (SVM)

We implemented SVMs as a second binary classification method, using C-Support Vector Classification in the scikit-learn library [5]. We chose to use SVMs so that we can experiment with different types of transformations to the data using kernels. In comparison to logistic regression, this would allow us to model a nonlinear decision boundary. We began by standardizing non-categorical features, like we did in logistic regression. Then we used grid search with group cross-validation to find the optimal hyperparameters for the model. Using group cross-validation ensured that samples that came from the same patient were not separated between the train and validation sets. We tested multiple types of SVM kernels: polynomial, sigmoid, linear, and rbf, as well as a range of other hyperparameters such as regularization parameters and gammas.

Once we identified hyperparameters to use, we evaluated the resulting model by training it on the entire training set, and calculating its accuracy on the held-out test set, as well as plotting the model's receiver operating characteristic curve.

In order to characterize the importance of different types of features on our model, we trained the model separately on the symptom features, age and sex features, and sound features, recording the resulting accuracy.

Fully Connected Neural Network

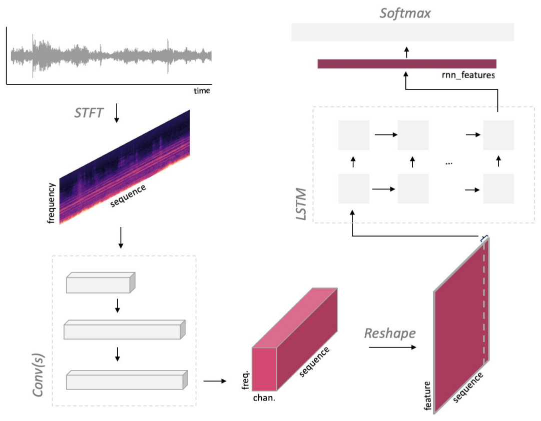
A simple fully connected neural network was implemented using the *PyTorch* library [11]. The architecture design parameters were: a number of hidden layers, a number of hidden units, activation function between layers, and dropout probability. We used a simple black box optimizer called *Gaussian Process Bandits* to determine the design of the neural network based on the aforementioned training set. The final design contains a single hidden layer, 28 hidden units in each hidden layer, ReLU activation function between layers, and a dropout probability of 12.89% between layers. The network was optimized using the *Cross-Entropy Loss* and *Adam optimizer* with default parameters. Further optimization of the architecture design is recommended when more data is available.

Temporal CRNN (TCRNN)

This model differs from the previous models in that the input is the full cough sounds themselves, instead of hand-engineered features, thereby preserving much of the information in the data. In order to investigate the feasibility of detecting COVID-19 from cough sound itself, we experimented with a classification model that only took the cough sound as input, with no additional features. In particular, the input was the raw .wav file of the cough segment of each audio file. This input was fed into a Temporal Convolutional Recurrent Neural Network, termed TCRNN with a combined CNN and LSTM architecture, outputting a binary classification of COVID-19 positive or negative. This temporal structure enables variable length input, which is useful for our task.

I. Spectrogram and STFT features

In more detail, the data is first transformed into a spectrogram, a temporal sequence of spectra. As in images, neighboring spectrogram bins of natural sounds in time and frequency are correlated; but in sound production, so are harmonics, or frequencies that are multiples of the same base frequency. Thus we can add a third dimension with the magnitudes of the harmonic series so that spatially local models such as a CNN can take these into account [13]. Therefore, we first obtain the short-time Fourier transform (STFT) of the signal to calculate the spectrogram, which serves as the features for our model.



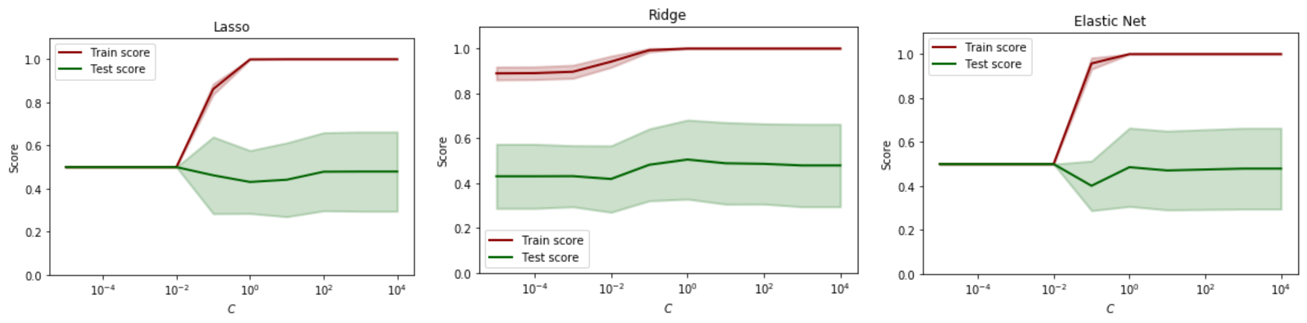
II. CRNN

CNNs and RNNs both have their respective advantages and disadvantages in audio classification. CNNs have a fixed receptive field, which can be limiting but also modified, while RNNs can in theory utilize an unlimited temporal context, but in practice may require modifications to achieve this. Ideally, CRNNs offer the best of both words by using the convolutional layers to extract local information, and the recurrent layers to combine it over a longer temporal context. This classification model employs an LSTM to better capture long term temporal dependencies. In summary, the CNN takes the spectrogram as input and consists of a sequence of 2D convolutions, followed by a bi-directional LSTM. The full model is described here (https://github.com/ksanjeevan/crnn-audio-classification#models) with the modification of removing a maxpool layer to accommodate our data size.

4. Results

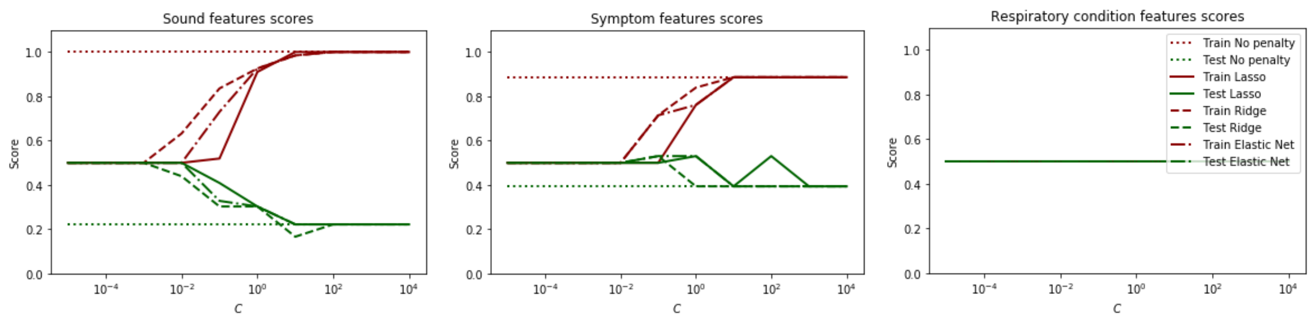
Logistic regression

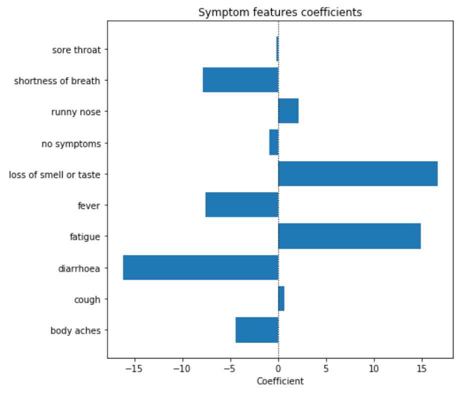
According to the cross-validation plots, the performance of the model on the full feature set is poor; the best average AUC score across the folds is achieved for ridge regression with *C = 1* and equal to 0.51 (which is almost a random coin flip). Note also that the 1SE intervals for the cross validation score are very wide (for the best model it is 0.51 +/- 0.2) indicating the significant instability in the estimated test score and strong dependence of the model on the train/test split.



Further, we fit the model for each group of features separately. First, we run logistic regression (without penalty) for age+sex as predictors and get test AUC = 0.83. Next we test sound, symptoms and respiratory condition features plotting the train/test score for different hyperparameter values. Notice that for respiratory condition features the AUC is equal to 0.5

regardless of the hyperparameter values, which can be explained by the fact that people with present respiratory conditions are strongly underrepresented. Moreover, the sound feature AUC scores do not exceed 0.5 for any type of penalty which implies that logistic regression is not able to distinguish ill and healthy patients using sound features only. Finally, the best test AUC value for symptom features is 0.53 for lasso penalty with *C = 100*.

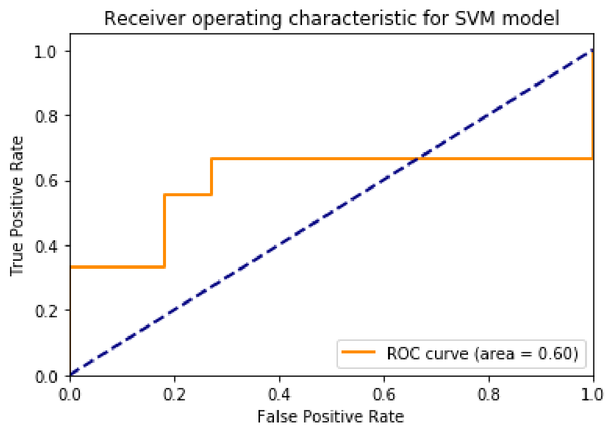




We conclude that general info features have the most predictive power, symptom features have moderate predictive power, and sound and respiratory condition features have no predictive power. We plot the lasso coefficients for the model with symptoms only; the most important symptoms are diarrhea, loss of smell or taste and fatigue.

Support Vector Machines

Through grid search, we found that the optimal hyperparameters for our data include an rbf kernel, a regularization parameter ‘C’ of 2.1, and a ‘gamma’ parameter of ‘auto’. During cross-validation, the model with these parameters achieved an accuracy of 0.59. However, on the test set, it achieved an accuracy of 0.5, meaning that its accuracy was only as good as random. Plotting its ROC curve below, we can see that the AUC was 0.6. Other metrics are reported in the table at the end of the section.



Next, we evaluated the model accuracy after training it on symptoms only, age and sex only, and sound features only. The accuracies achieved for each of the three categories, respectively, was 0.3, 0.7, and 0.3. Thus, our model suggests that age and sex alone are more predictive than any of the other features, and, like with logistic regression, that our sound features probably do not contain much useful information for the model.

Neural networks

The model was trained on the preprocessed features and evaluated on the test set. The results are presented in the table below. The accuracy of 0.7 suggests that it is likely that the model optimized for age and sex alone, as noticed in previous sections, and that we have reached the limit of extractable information from the preprocessed features.

To account for this, we have returned to the raw and cropped cough files and trained a Temporal Convolutional Recurrent Neural Network. Performing 5-fold cross validation, we averaged the metrics in each of the 5 runs to obtain the TCRNN performance with an accuracy of 58%, precision of 0.55, recall of 0.52 and f1-score of 0.53.

The following table presents a comparison of our classification models with the metrics of accuracy, precision, recall and f1-score. Our classes are relatively balanced, but we still report macro-average for precision, recall and f1-score.

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**
SVM	0.50	0.52	0.52	0.49
Fully Connected NN	0.70	0.67	0.67	0.67
TCRNN	0.58	0.55	0.52	0.53

5. Discussion and Conclusion

Result Analysis

The key takeaways from our results are:

- Logistic regression and SVM show no predictive power of preprocessed sound features,
- Age and sex are moderately predictive given the dataset,
- The fully connected neural network which accounted for the accuracy of ~70% has reached the limit of extractable information from the preprocessed features. It implies that feature preprocessing has resulted in the significant loss of information prohibiting successful classification of the recording,
- We posit that using raw cough audio files or at least maintaining some form of temporal information in the features is key to classify the recording,
- Our results from the TCRNN demonstrate potential promise on the feasibility of using cough sound to detect COVID-19. In particular, we achieved ~58% accuracy on 5-fold validation on just .wav alone, which shows there may be some signal in the cough sounds. The model is able to overfit on the small training data which signifies that it is complex enough for our small dataset and more data and experimentation are required to fully determine the efficacy of this method. Therefore, we tentatively posit that there is signal in the cough sound for detecting COVID-19, while stating that this is only an initial

investigation and we need to thoroughly vet our process and perform analysis on a more substantial dataset. In addition, minimal tuning and data augmentation was performed, so there could be improvements in performance in this area.

Conclusion

In summary, we have provided a pipeline for processing the audio recordings, segmenting the cough sound, conducting signal processing, extracting features and classifying the presence of COVID-19. We focused on breadth and explored several possibilities for most of these components, especially the classification task. One of our main challenges while working on this project was only having access to a very small data set. The small number of training examples and lack of a diverse set of examples led to high variation in the results. Furthermore, we believe that careful feature extraction is essential to successful classification, as averaging of the signal processing features over the time domain resulted in a loss of important data. Potential for future work includes:

- Re-evaluation of classification models with more data
- Augment the raw cough audio files with preprocessed features and model the classification task using the recurrent neural network,
- Perform confounder analysis (age and sex). It is necessary to decide whether these are features we want to aid our prediction or biases we want to filter out. A potential exploration is modeling linear confounders with a GLM and regressing out biases.
- More in-depth analysis of the TCRNN model along with more tuning and analysis.
- Extension from binary classification to adding another class for unknowns, or outputting a confidence score.

In conclusion, there are several intentional choices to be made in the task of detecting COVID-19 from cough sounds. A preliminary prediction from this work is that cough sounds may not have high enough accuracy to replace testing, but can potentially serve as a helpful triage or diagnostic tool.

Acknowledgements

We would like to show our gratitude to Jigar Doshi, Piyush Bahad and other members of Wadhwani Institute of Artificial Intelligence, and the teaching staff of CS472 for organizing the class, sharing the cough datasets, Amazon AWS resources and providing feedback during our weekly meetings.

References

[1] R. Pramono, S. Imtiaz and E. Rodriguez-Villegas, “A Cough-Based Algorithm for Automatic Diagnosis of Pertussis”, *PLoS One*, 2016, 11(9), doi:10.1371/journal.pone.0162128

[2] Y. Amrulloh, U. Abeyratne, V. Swarnkar and R. Triasih, "Cough Sound Analysis for Pneumonia and Asthma Classification in Pediatric Population," *2015 6th International Conference on Intelligent Systems, Modelling and Simulation*, Kuala Lumpur, 2015, pp. 127-131, doi: 10.1109/ISMS.2015.41.

[3] Y. Amrulloh, D. Wati, F. Pratiwi and R. Triasih, "A novel method for wet/dry cough classification in pediatric population," *2016 IEEE Region 10 Symposium (TENSYMP)*, Bali, 2016, pp. 125-129, doi: 10.1109/TENCONSpring.2016.7519390.

[4] H. Chatrzarrin, A. Arcelus, R. Goubran and F. Knoefel, "Feature extraction for the differentiation of dry and wet cough sounds," *2011 IEEE International Symposium on Medical Measurements and Applications*, Bari, 2011, pp. 162-166, doi: 10.1109/MeMeA.2011.5966670.

[5] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. *Scikit-learn: Machine Learning in Python.* J. Mach. Learn. Res. 12, null (2/1/2011), 2825–2830.

[6] Laput, G., Ahuja, K., Goel, M. and Harrison, C., 2018, October. Ubicoustics: Plug-and-play acoustic activity recognition. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (pp. 213-224).

[7] Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B. and Vijayanarasimhan, S., 2016. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*.

[8] Jort F. Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. 2017. Audio Set: An ontology and human-labeled dataset for audio events. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP), pp. 776-780. IEEE.

[9] Frederic Font, Gerard Roma, and Xavier Serra. 2013. Freesound technical demo. In Proceedings of the 21st ACM international conference on Multimedia (MM '13). ACM, New York, NY, USA, 411-412. DOI: https://doi.org/10.1145/2502081.2502245

[10] Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J. and van der Walt, S.J., 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods*, *17*(3), pp.261-272.

[11] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. and Desmaison, A., 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems* (pp. 8024-8035).

[12] Imran, A., Posokhova, I., Qureshi, H.N., Masood, U., Riaz, S., Ali, K., John, C.N. and Nabeel, M., 2020. AI4COVID-19: AI enabled preliminary diagnosis for COVID-19 from cough samples via an app. *arXiv preprint arXiv:2004.01275*.

[13] Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuoyiin Chang, Tara Sainath. Deep Learning for Audio Signal Processing. In Journal of Selected Topics of Signal Processing, Vol. 13, No. 2, May 2019, pages 206–219.

Websites accessed

[1] https://www.kdnuggets.com/2020/02/audio-data-analysis-deep-learning-python

[2] https://towardsdatascience.com/music-genre-classification-with-python

[3] https://hackernoon.com/how-to-apply-machine-learning-and-deep-learning-methods-to-audio-analyis

[4] https://timsainburg.com/noise-reduction-python.html

[5] https://towardsdatascience.com/a-guide-to-svm-parameter-tuning-8bfe6b8a452c

[6] WHO Coronavirus Disease (COVID-19) Dashboard https://covid19.who.int