COVIDector: A Federated Learning Approach to Predicting COVID-19 Status

Background

Clinical Problem

Just as there was a sure in the number of COVID-19 cases throughout the world, so too did the technological community step up to work on providing solutions for the pandemic. A major concern that still needs to be addressed, however, is how to predict which patients most likely have COVID-19 based on their demographics, symptoms, comorbidities, etc. Being able to model this association would not only help patients in feeling more secure, but also it would alleviate a great deal of clinician burden for managing cases and testing for individuals who eventually test negative, or who come in later during the disease progression than would be preferred. [1]

Current Approaches

Currently, most prediction models that exist are geared toward understanding how mortality rates differ across regions of the world [2] or to figure out how the virus is traveling. [3]With regard to AI based solutions, there is skepticism as to whether AI is ready to tackle COVID-19 with regard to any application, especially prediction. [1,4,5] One particular application that seems to have the best prospects is with predicting status based on CT scans for patients, but the datasets are still fairly small compared to what the quantity needed for robust algorithms. Further, any big data approach that has currently been implemented relies heavily on patients who have been tested and arrive at clinics or hospitals for treatment. Though there has been some work done to monitor symptoms to determine risk, [6] as well as plenty of symptom tracking apps, there have not been applications that are open-source and could help supplement existing public datasets [7] with testing status information.

Approach

As such, I propose *COVIDector*, a machine-learning tool that can allow us to understand the underlying relationship between relevant health information (symptoms, demographic information, social determinants of health, etc.) and testing status. In its initial iteration, COVIDector will serve as a binary classification tool to predict with some certainty whether someone has COVID-19. It will serve as a component of *CoronaTracker*, [8] an open-source application designed to monitor and educate the public while preserving their privacy, which is why a major component of COVIDector will be an emphasis on preserving privacy while still being able to perform predictions. If this proves to be successful, the rate of meaningful testing for COVID-19 will increase and it should reduce the cognitive burden on patients and providers who are fearful of the prevalence and impact of COVID-19 that is still to come.

Methods

The three main methods I have incorporated into this project are: logistic regression (for baseline assessments), feedforward neural network (to better understand the nonlinearities and feature importance) and federated learning (accomplished by using the PySyft library).

Logistic Regression (Baseline)

Logistic regression is a common baseline metric for binary classification problems because it allows the researcher to assess the extent to which linear models can represent the targeted classes. It is the simplest form of a neural network consisting of only one neuron that accepts inputs, applies a linear function, and then passes that into a non-linear activation function, generally the Rectified Linear Unit (ReLU) to be represented as either a 0 or 1.

Feedforward Neural Network (Main Classifying Tool)

Feedforward neural networks (FFNN) are simple and common deep learning tools used for classification issues because they do not have as much complexity as their convoluted and recurrent counterparts. They serve as the extension of logistic regression models because they allow for the assessment of nonlinearities in the data through a series of N neurons that feed directly into one another. They also allow researchers to easily bridge into using multidimensional and n-nary data from the binary classification case.

Federated Learning (PySyft)

Federated Learning is an approach to privacy in deep learning that seeks to distribute models out to users to collect and train rather than to bring the data to the server itself (Fig. 1). [9] It provides the user with conventional implementation of the PyTorch syntax with a comprehensive API. Generally, the algorithms run about twice as long with PySyft than they do with regular PyTorch.

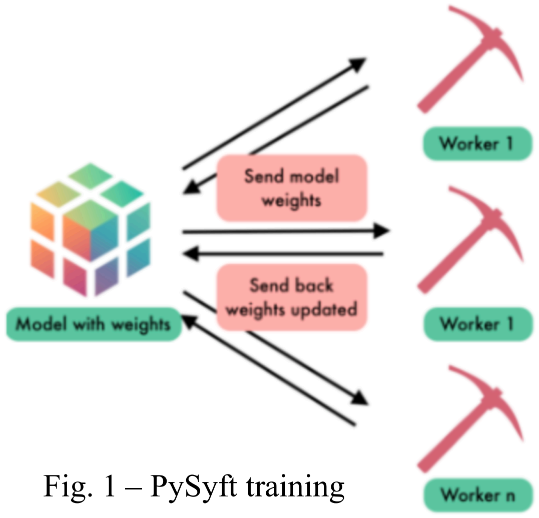


Fig. 1 – PySyft training

Data

The dataset that I am using was obtained from Kaggle [7] and it consists of symptoms along with their severities, as well as age and exposure, for 316,800 individuals across the globe (Table 1). This dataset, however, did not include any testing results associated with each of the entries so I used simulated the data using uniform

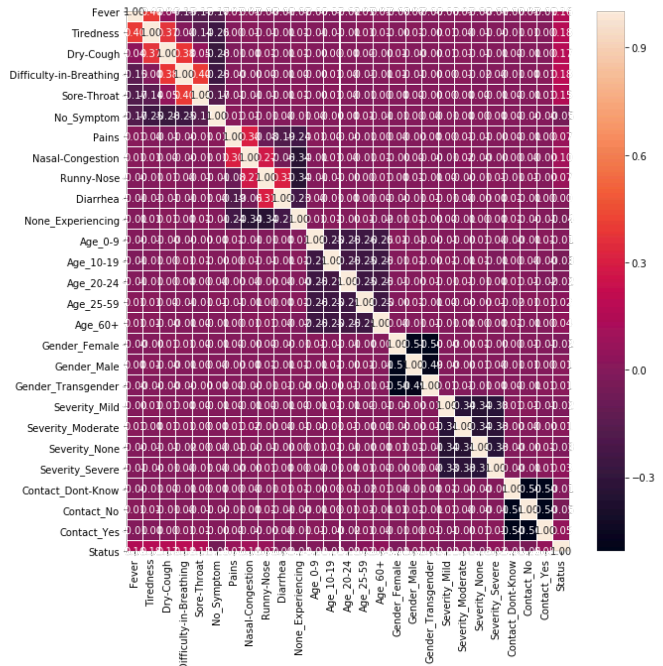
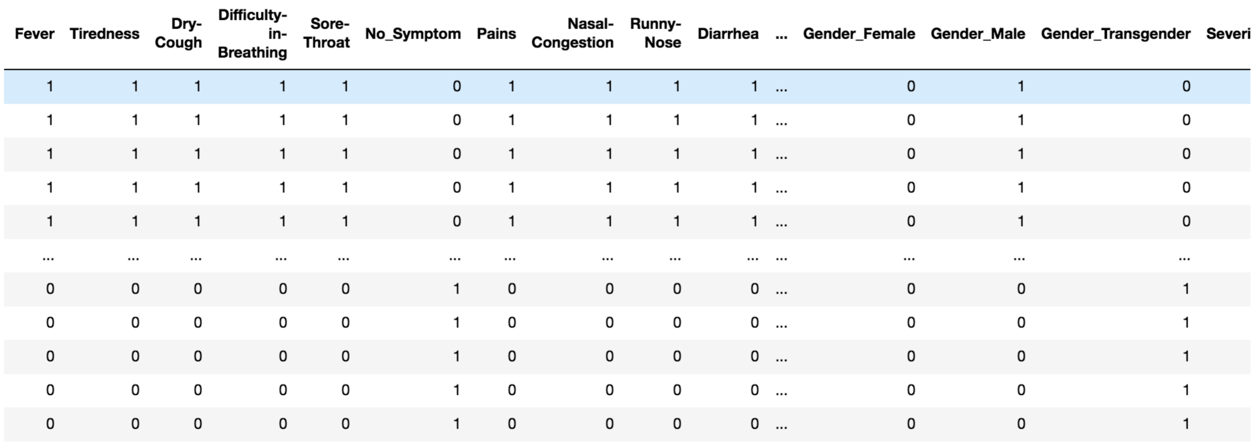


Figure 2 – Heat map of Symptoms and Status Data

distributions with weighting probabilities for the testing status that were associated with general prevalence of particular symptoms over others globally. Since,c I do not have much experience with data simulation, and there are a wide number of features, most of the data did not have a high correlation with the testing status (Fig. 2). As was expected, to try and reflect the global infection rate, the dataset was unbalanced, so I used the PyTorch weighted random sampling approach to remove any imbalance in favor of the majority class, i.e., COVID- patients. I randomly sampled on 20,000 data entries for the sake of memory in the simulation and evaluated it through mini-batching.



Fever	Tiredness	Dry-Cough	Difficulty-in-Breathing	Sore-Throat	No_Symptom	Pains	Nasal-Congestion	Runny-Nose	Diarrhea	...	Gender_Female	Gender_Male	Gender_Transgender	Severi
1	1	1	1	1	0	1	1	1	1	...	0	1	0	
1	1	1	1	1	0	1	1	1	1	...	0	1	0	
1	1	1	1	1	0	1	1	1	1	...	0	1	0	
1	1	1	1	1	0	1	1	1	1	...	0	1	0	
1	1	1	1	1	0	1	1	1	1	...	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
0	0	0	0	0	1	0	0	0	0	...	0	0	1	
0	0	0	0	0	1	0	0	0	0	...	0	0	1	
0	0	0	0	0	1	0	0	0	0	...	0	0	1	
0	0	0	0	0	1	0	0	0	0	...	0	0	1	
0	0	0	0	0	1	0	0	0	0	...	0	0	1	

Table 1 – Symptoms Data

Results

There are currently no viable or informative results that can be determined from my models on account of the data processing. I ran the models, along with their respective optimizers and loss functions on MNIST data provided by PyTorch with standard results, decreasing loss, increasing accuracy $\geq 90\%$, steady convergence of training and validation loss, etc. When I run the logistic regression model and FFNN models on the COVID-19 data, they overfit with accuracy generally in the range of 15-30% (Fig. 3a), regardless of layer type, activation function, learning rate, or dropout. Increasing the batch size above 50 can help "improve accuracy," but is generally discouraged. [10] Further, I created ROC curves for each of the models and they are characteristic of overfit classifiers in that the AUROC is too close to one for the loss and accuracy results provided earlier (Fig. 3b).

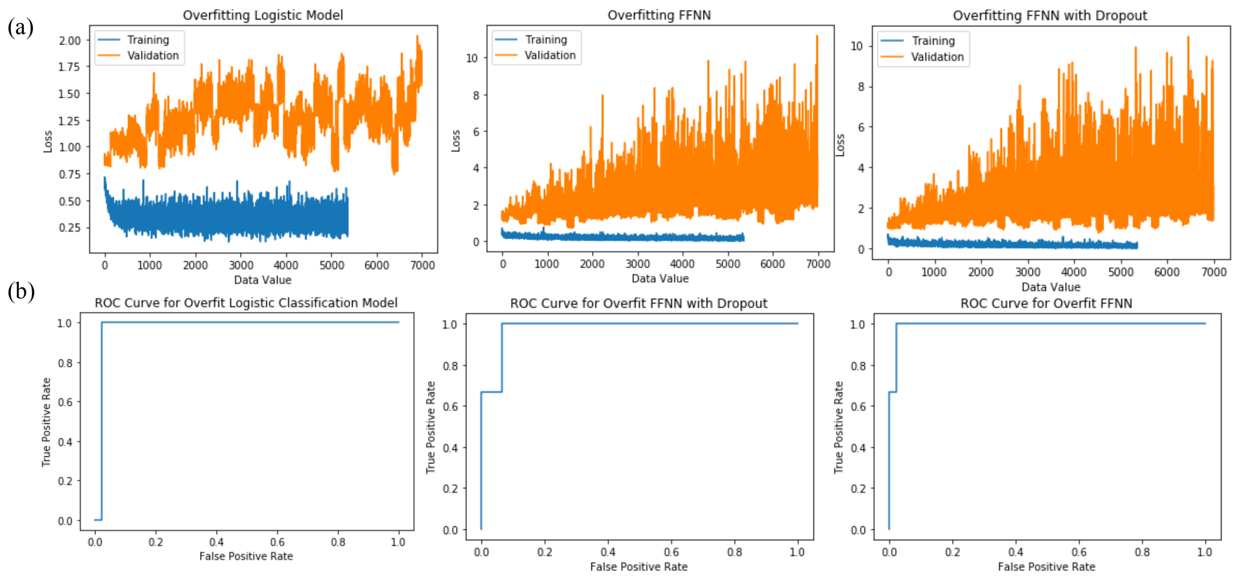


Fig.3 (a) Plots of training versus validation loss across models with respective ROC curves (b)

Discussions/Conclusions

My hope is that I can work on this further for use within CoronaTracker that is not hampered by any of the difficulties that I may have introduced into the code. Particularly, I would like it to be possible for FFNNs to be a good starting point for classifying COVID-19 status because they appear to be successful within the scope of PySyft for generating reliable classification models (when I can incorporate one of my own). [11] It is possible that some of this could be contributed to how I simulated the data or even that I had to do so in the first place. Another option could be to shift from mini-batch gradient descent to batch gradient descent, since that seems to increase my accuracy and minimize the variance in the validation loss (though it is not very stable). Further, I am considering time series/survival analysis (possibly using LSTM instead), comparing NN to non-NN solutions, if there are encryption-like wrappers/libraries, and including NLP of free-text.

References

[1] B. McCall, “COVID-19 and artificial intelligence: protecting health-care workers and curbing the spread,” *Lancet Digit. Health*, vol. 2, no. 4, pp. e166–e167, Apr. 2020, doi: 10.1016/S2589-7500(20)30054-6.

[2] F. A. B. Hamzah *et al.*, “CoronaTracker: World-wide COVID-19 Outbreak Data Analysis and Prediction,” nCoV, preprint, Mar. 2020. doi: 10.2471/BLT.20.255695.

[3] M. N. Kamel Boulos and E. M. Geraghty, “Geographical tracking and mapping of coronavirus disease COVID-19/severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) epidemic and associated events around the world: how 21st century GIS technologies are supporting the global fight against outbreaks and epidemics,” *Int. J. Health Geogr.*, vol. 19, no. 1, p. 8, Mar. 2020, doi: 10.1186/s12942-020-00202-8.

[4] W. Naudé, “Artificial intelligence vs COVID-19: limitations, constraints and pitfalls,” *Ai Soc.*, pp. 1–5, Apr. 2020, doi: 10.1007/s00146-020-00978-0.

[5] K. C. Santosh, “AI-Driven Tools for Coronavirus Outbreak: Need of Active Learning and Cross-Population Train/Test Models on Multitudinal/Multimodal Data,” *J. Med. Syst.*, vol. 44, no. 5, 2020, doi: 10.1007/s10916-020-01562-1.

[6] C. J. Wang, C. Y. Ng, and R. H. Brook, “Response to COVID-19 in Taiwan: Big Data Analytics, New Technology, and Proactive Testing,” *JAMA*, Mar. 2020, doi: 10.1001/jama.2020.3151.

[7] “COVID-19 Symptoms Checker.” https://kaggle.com/iamhungundji/covid19-symptoms-checker (accessed May 05, 2020).

[8] “CoronaTracker,” *CoronaTracker*. https://coronatracker.me (accessed Apr. 01, 2020).

[9] T. Ryffel *et al.*, “A generic framework for privacy preserving deep learning,” *ArXiv181104017 Cs Stat*, Nov. 2018, Accessed: May 05, 2020. [Online]. Available: http://arxiv.org/abs/1811.04017.

[10] D. Masters and C. Luschi, “Revisiting Small Batch Training for Deep Neural Networks,” *ArXiv180407612 Cs Stat*, Apr. 2018, Accessed: May 05, 2020. [Online]. Available: http://arxiv.org/abs/1804.07612.

[11] “Encrypted Deep Learning Classification with PyTorch + PySyft,” *OpenMined Blog*, Apr. 16, 2019. https://blog.openmined.org/encrypted-deep-learning-classification-with-pysyft/ (accessed May 05, 2020).