COVID-19 Infection Detection Using Machine Learning

Leo Wang and Haiying Shen
Department of Computer Science
University of Virginia
Charlottesville, VA 22904
{yw7uc, hs6ms}@virginia.edu

Kyle Enfield and Karen Rheuban
School of Medicine
University of Virginia
Charlottesville, VA 22904
{ke4z, ksr5g}@hscmail.mcc.virginia.edu

2021 IEEE International Conference on Big Data (Big Data) | 978-1-6654-3902-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigData52589.2021.9671700

***Abstract*—The COVID-19 pandemic is an ongoing pandemic of coronavirus disease since 2019. Millions of cases and deaths attributed to it have been confirmed in the world. So far the detection of COVID-19 heavily relies on the specialized tests (e.g., based on saliva or respiratory swabs). Some approaches use smart devices (e.g., Whoop) for coronavirus infection detection using respiratory rate. Machine learning (ML) techniques have become a promising approach for the coronavirus infection detection. Therefore, in this paper, we introduce a machine learning based COVID infection predictor. We measure the prediction accuracy of five ML models. We use Chi-square test and knowledge-based manual feature selection to select important features for prediction to reduce prediction time overhead without compromising prediction accuracy. We also study the accuracy with different input features (those that can be measured by medical devices and by smart devices) and find that removing some features has no or slight influence on the prediction accuracy. Since insufficient or unbalanced training data decreases the prediction accuracy, we further propose a Generative Adversarial Network (GAN) ML based predictor that produces synthetic data (close to real data) for ML training. Our extensive experiments show the effectiveness of our methods in improving the detection accuracy. Our study results can provide guidance on developing the coronavirus infection predictors based on different data sources and devices. We open sourced our code in GitHub.**

I. Introduction

The COVID-19 stands for coronavirus disease 2019. It is a serious disease caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), and it ranges from no symptoms or mild ones to respiratory failure and death [1]. By May 12, 2021, 160M cases and 3.32M deaths of COVID-19 have been confirmed in the ongoing global pandemic. The United States has become the country that have conducted most COVID tests and confirmed most cases and deaths [2]. Performing COVID tests on large population is an expensive and time-consuming task. It usually brings extra costs to both health departments and patients. Besides, the wide range of symptoms covers several common ones, including fever, chills, cough, headache, and so on. Watching for symptoms is not always effective to timely detect COVID-19 since these symptoms may appear 2 to 14 days after exposure to the virus [3]. In this case, it will be greatly helpful to have a stable predictor of COVID-19 test results based on only patients' health data, such as systolic blood pressure, pulse oxygen saturation, etc., that can be measured by medical devices or heart rate, respiratory rate, etc., that can be measured by smart devices. We hope that a reliable predictor will be able to lead to better understanding of the COVID-19 symptoms and support the current COVID-19 testing infrastructure and protection measures.

In this paper, we will study the accuracy of different machine learning (ML) models to conduct coronavirus infection prediction based on the COVID-19 related patient health data. The machine learning models include Logistic Regression (LR) [4], Bagging Classifier (BC) [5], Balanced Bagging Classifier (BBC) [6], Gradient Boosting Classifier (GBC) [7], and XGBoost Classifier (XGB) [8]. We will test these models, evaluate their performances, and select the best one to be our predictor. We also introduce two ML input feature selection methods: Chi-square test [9] and knowledge-based manual feature selection, to remove unnecessary input features and improve predictor performance. Considering that some smart devices cannot sense certain features, we gradually remove features from the input and study how the prediction accuracy will change. Moreover, we solve the problem that the lack of training data and the imbalanced feature in the data (with much fewer COVID-19 infection positive cases) may affect the accuracy of the trained ML model. Specifically, we will use Generative Adversarial Network [10] to create synthetic data that is close to real data for training. The network will fit the same dataset and be able to generate more similar data.

We summarize our observations from our measurement study below:

- The Gradient Boosting Classifier is the model with the best accuracy performance, and the XGBoost Classifier also achie-ves comparable results. The Logistic Regression, the Bagging Classifier, and the Balanced Bagging Classifier are able to achieve good results.
- Not all the features in the COVID-19 related patient health data are important or useful for the infection detection. The Chi-square test and manual feature selection based on knowledge are useful to select important input features and remove unimportant features without compromising prediction accuracy. Removing some features will slightly decrease the infection detection accuracy. Thus, the input features can be adaptively selected based

on the availability of the features and the computation time limit.

- The original dataset is greatly skewed, and the GAN is able to generate balanced dataset. All models' accuracy can be significantly improved using the balanced dataset (instead of the skewed dataset) for training.

We open sourced our code of all the ML classifier models and feature selection methods. [1] The study is beneficial in several ways. It not only builds the COVID-19 test results predictor, but also analyzes the pattern of the patients' data and highlight important features to make future detection easier. The data generation task performed by neural networks will be capable of handling the issues caused by small or unbalanced dataset in similar studies. After reaching a reliable level of performance, the final selected predictor can still be further studied and improved with the new collected COVID-19 patients dataset.

The rest of the paper is organized as follows. Section II presents the related work. Section III introduces our dataset and data preprocessing methods. Section IV describes our machine learning approaches. Section V presents the performance evaluation. Finally, Section VI concludes the paper with remarks on our future work.

II. Related Work

Since the outbreak of the COVID-19 pandemic in early 2020, there have been a lot of existing work on the COVID-19 detection and prediction using machine learning techniques. Earliest work mostly relied on the clinical data collected in Wuhan, where the first COVID-19 case was reported in China. Yan *et al.* [11] built machine learning tools using a database of blood samples from 404 infected patients in the region of Wuhan. The tools were then used to select three biomarkers and predict the survival of individual patients. The usage of blood samples was shown to be effective and adopted by some other researchers. Wu *et al.* [12] in their study extracted 11 key blood indices from 49 clinical available blood test data and built the final assistant discrimination tool using the random forest algorithm. Brinati et al. [13] developed two machine learning classification models using hematochemical values from routine blood exams drawn from 279 patients, who were admitted to the San Raffaele Hospital (Milan, Italy) emergency-room with COVID-19 symptoms. Yao *et al.* [14] trained a support vector machine model using 28 features extracted from the blood/urine test data. The model could then be used as a severeness detection model of COVID-19.

A lot of research has been also conducted with various datasets to assess and predict the risk of severe illness from COVID-19. Assaf *et al.* [15] used machine learning models to predict patient risk for critical COVID-19 based on their status at admission. Heldt *et al.* [16] analyzed data of 879 confirmed SARS-CoV-2 positive patients admitted to a two-site NHS Trust hospital in London, England and applied multivariate logistic regression, random forest and extreme gradient boosted trees in order to evaluate the risks of these patients. Schning *et al.* [17] developed models to calculate clinical risk scores, which could assist in automated early identification of severe clinical outcomes. Prediction works also cover emergency cases. The study of de Moraes Batista *et al.* [18] developed predictors that only result from emergency care admission exams. The work of Cheng *et al.* [19] focused on ICU transfers, which usually require a series of operations in short time, and developed a machine learning-based risk prioritization tool that could predict the transfer within twenty-four hours. Besides, effective models were helpful not only for patients but also for non-patients. The models were used for separating positive cases from negative ones and reducing the chances of infections. Awal *et al.* [20] in the study proposed a machine learning-based framework for quick identification of COVID-19 infection. The framework used Bayesian optimization to optimize the hyperparameters of the classifier and ADAptive SYNthetic (ADASYN) algorithm to balance the COVID and non-COVID classes of the dataset. Vaid, Cakan, and Bhandari [21] developed a model to uncover hidden patterns based on reported cases and to predict potential infections in North America. The discoveries may allow us to properly and wisely estimate the spread of the COVID-19.

Several efforts used physiological metrics to detect illness infection. Scripps Research Institute [22] studied the correlating changes in heart rate (HR) to influenza. One study assessed HR, activity, and sleep data to identify influenza-like illness via smart devices [23]. Scripps collaborated with Stanford University and Fitbit in assessing whether changes in HR, skin temperature, and SpO2 could be used to predict the onset of COVID-19 before symptoms start [24]. Whoop has focused on correlating changes in respiratory rate and recovery levels to predicting COVID-19 [25]. Researchers from Duke University monitored an individual's HR, sleep schedules, oxygen levels, and activity levels by using devices such as Fitbit and data from the Apple Health app to conduct COVID-19 infection detection [26], [27].

III. Dataset and Data Preprocessing

A. Dataset Introduction

The dataset we use in this study is a dataset of COVID-19 related patients' data collected from an inpatient department from a hospital. The dataset contains 7380 patients and each patient has 41 columns (i.e., features) in total. The features include patients' basic information, such as age, gender, and race. The features also include patients' health and medical data, including height, weight, heart rate, systolic blood pressure (sbp), diastolic blood pressure (dbp), pulse oximeter (pulse ox) using SpO2 sensor, respiratory rate (resp rate), amount of glucose (cmp glucose). Also, a feature in the dataset stores the COVID-19 test results of each patient. Most features contains numerical values, but still some of them are stored in the format of string. Many of these features are important data sources to build our predictor.

[1]Our source code research artifacts are available to download from: https://github.com/yw7uc/COVID-19-Infection-Detection-Using-Machine-Learning.

Missing value detection is necessary for data preparation, as empty or invalid values usually could not be used as legal inputs for models and algorithms. Also, it provides an overview on the number of original data points, and this may become an important reference in the final evaluation of the outcomes. Given the dataset contains 7380 rows, we notice that there exist some missing values in most columns. Figure 1 records the count of each feature in this dataset. The missing data needs to be handled during the data preprocessing step.

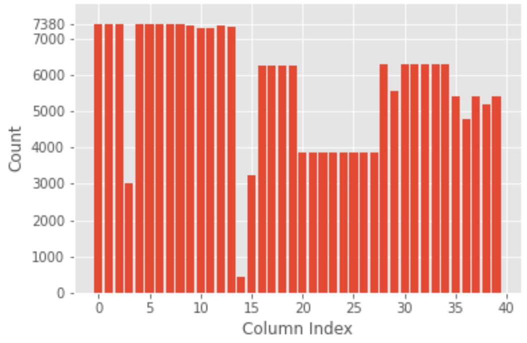

Fig. 1. Non-null Values Count.

For the data distribution, we only focus on the column with the name COVIDResult, since it is the label column in the dataset. We realize the distribution is an important information because a dataset with skewed distribution misleads the models and result in inaccurate predictions. To find out the distribution, we similarly use a bar chart, as shown in Figure 2, to plot the count values in the label column. From the figure, we notice that the dataset is greatly unbalanced and skewed to the negative test results. Among all 7380 values, there are only 470 positive cases, which account for merely 6.37% of the dataset.

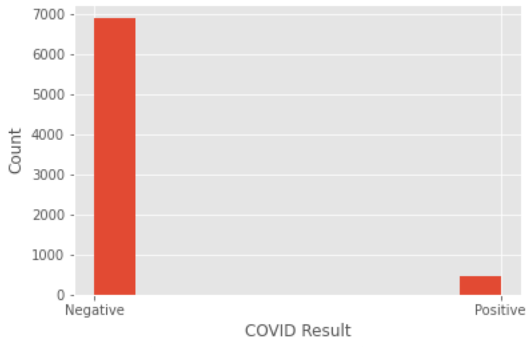

Fig. 2. COVID Result Distribution.

B. Data Preprocessing

The data preprocessing step begins with splitting the dataset into train and test sets. We generate the train set with 80% data from the original dataset and the test set with 20%. Both are generated through random data selection and follow the exactly same preprocessing procedures described below. When working with classifier models we use the train set for model training and the test set for performance evaluation with different metrics scores. Next we design two different preprocessing plans separately for numerical and categorical values. For numerical columns, we begin with handling the missing values detected in the data analysis step, and selected a Simple Imputer tool [28] for this task. The Imputer is configured to look for NaN values and replace them with 0. We do not choose the mean nor median strategy in order to avoid manually adding weights to some non-zero values. Next, to eliminate the impacts of different value ranges on our training results, a Normalizer [29] is included to apply L2 normalization to the data. Compared to Standard Scaler [30], which standardizes features by removing the mean and scaling to unit variance, Normalizer is able to avoid generating negative values, which are invalid inputs for some of the models. The categorical columns include the label column and some feature columns. To transform these values from strings to non-negative integers, we use two kinds of encoder: Ordinal Encoder [31] for feature columns and Label Encoder [32] for the label column. Ordinal Encoder transforms features into an array-like of integers, while Label Encoder works well for boolean values in the label column. In the end we merge the numerical and categorical preprocessed data into one prepared dataset, and we use it for model training and testing.

IV. Machine Learning Approaches

In this paper, we use five machine learning approaches for the infection detection. We also introduce two feature selection methods on the input dataset and analyze their effectiveness on improving the performance of our classification models. To deal with the problem of low prediction accuracy due to the unbalanced data, we further build a Generative Adversarial Network to create a synthetic data that is close to the real data for training.

Physiological metrics are effective in the detection even before symptoms arise. Many metrics derived from heart rhythm such as heart rate (HR), heart rate variability (HRV), resting heart rate (RHR), and respiration rate (RR) could serve as potential markers of COVID-19 infection and are already can be measured by wearable devices such as the Apple Watch, Whoop Strap, Fitbit, Zephyr BioHarness, or VivaLNK Vital Scout. The papers in [33], [34] provide lists of commercial wearable sensors used to monitor physiological parameters necessary for COVID-19 detection and current clinical trials utilizing commercial wearable sensor devices to diagnose and monitor COVID-19. Some smart devices cannot sense certain features such as CMP Glucose. Then, a question is: when some features in the dataset from the hospital are not available, what will be the machine learning based prediction accuracy? Therefore, we gradually remove the input features and measure the prediction accuracy. The results will give guidance on developing machine learning based infection prediction methods either in the hospital to assist doctors or on smart devices for their wide adoption on non-patients.

In the following, we first introduce our input feature selection and the five machine learning approaches, and then present our measurement for these approaches. Second, we introduce GAN and then present the measurement results for the five machine learning approaches trained using the data generated by GAN.

A. Input Feature Selection

Feature selection is one of our strategies to improve the performance. Classifiers are expected to make better and more reliable predictions after eliminating some less relevant features. We use two feature selection methods on our dataset. The first one uses the Chi-squared test to select top features from the dataset. Given a dataset with labels and features, it computes Chi-squared statistics between each non-negative feature and class. These statistics represent the dependence between the label column and every feature. We use this method to return the top 10, 20, and 30 features with the highest Chi-squared test scores. Classifiers are trained and tested again only on these selected features, and the results are compared with the original results to see if they are really improved.

The second method is designed based on the general knowledge among these patient features. To begin with, we narrow down to only twelve selected key features: Age, First Race, Ethnicity, Sex, Height, Weight, Heart Rate, Systolic Blood Pressure, Diastolic Blood Pressure, Pulse Oximetry, Respiratory rate, and CMP Glucose.

We use only these 12 features to train and test our models and compare the results to with the original results. We then gradually remove some features from the list following the procedure below:

1) Remove Systolic Blood Pressure and Diastolic Blood Pressure, ending up with 10 features.
2) Remove Pulse Oximetry and end up with 9 features.
3) Remove CMP Glucose and end up with 8 features.
4) Remove Respiratory rate and end up with 7 features.

The purpose of this process is to simulate the situation where such removed features cannot be measure by the devices. Also, by removing features, we could better understand the influence of individual features on the classifier. During this process, every classifier is trained and tested again after each removing step, and all the results are recorded and compared in the end to find out the feature combinations that allow the classifiers to produce the best outcomes for the prediction.

B. ML Models for Prediction

During the experiment, all the classifier models are implemented using the scikit learn API [35], [36] and imbalanced-learn API [37].

1) Logistic Regression: Our COVID test result prediction is regarded as a binary classification task, and thus we start from Logistic Regression, one of the most basic binary classifier, on the prepared dataset for the infection detection. The model uses a logistic function to model a binary dependent variable and hence performs the classification. For the model configuration, we use Limited-memory BFGS as the solver to perform the linear approximation. L2 penalty is used for regularization, and we use 500 for the number of maximum iterations to ensure that the solver converges. The regularization strength is represented by the C value, which is set to 1 to avoid over-fitting issue on this skewed dataset.

2) Bagging: In order to search for possible better outcomes, we apply Bootstrap Aggregating on our dataset as a machine learning ensemble meta-algorithm, which relies on ensemble classifiers to boost the performance of algorithms by using multiple copies of the same model on different subsets of the input training data. Compared to Logistic Regression, Bootstrap Aggregating gives us opportunity to train classifiers on samples instead of the whole data. This allows us to understand the difference between data queries and whether the difference is large enough to influence the model performances. Specifically, we use Bagging Classifier for performing this algorithm. It is the most basic and representative classifier of the Bootstrap Aggregating and allows us to customize the parameters of the model. Figure 3 shows the structure of our Bagging Classifier. It fits base classifiers each on random subsets of the original dataset and then aggregates their individual predictions through a voting procedure to form a final prediction. All the samples are random subsets drawn by the classifier with replacement based on a random state variable. The randomization and aggregation in this algorithm are effective to reduce the variance in results. In this Bagging Classifier, we pass the previous Logistic Regression classifier into the base_estimator parameter in order to keep using it for fitting the dataset. The number of this base estimator is then set to 10 to match the expected number of data subsets. After passing the patients' data into the model, ten random subsets are sampled and used to correspondingly train each base estimator. In the end, all ten results are then examined together, and the class that is predicted most often is returned as the output.

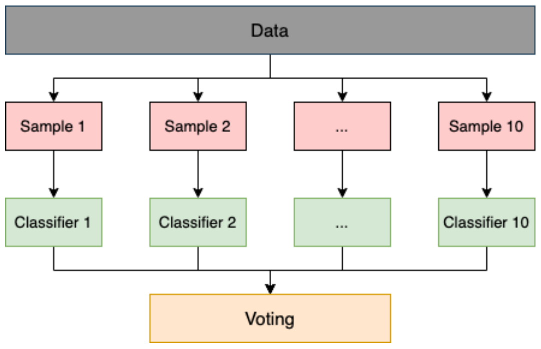

Fig. 3. Bagging Classifier Structure.

The second model we use for bagging algorithm is the Balanced Bagging Classifier. It is similar to the Bagging Classifier, but includes an additional step to balance all the randomly drawn samples. The goal of using this model is to see if this balancing step is able to reduce the negative effects brought by the skewed dataset. The sampling is controlled by the parameter sampler, and in our model we use the Random Under Sampler [38], which under-samples the majority class by randomly picking samples. The majority class is defined as the most frequent class, and in our dataset it is the positive class. The sampler is configured to pick samples with replacement in order to follow the procedures in the Bagging algorithm.

3) Boosting: Boosting is another ensemble meta-algorithm we use to train and test on our dataset, and we choose two models to perform this algorithm. Compared to Bagging, which combine the sample results only at the end, Boosting builds up the sample results throughout each iteration. This design trains the single classifier in multiple steps. By comparing the results of Boosting and Logistic Regression, we are able to understand the impact of the dataset size and the number of training iterations to the classifier model. The first model is the Gradient Boosting Classifier, a most typical classifier model of Boosting algorithm. Figure 4 shows its structure, which works in an iterative fashion and builds a single strong classifier based on multiple weak ones. Compared to the Bagging Classifier, the Gradient Boosting Classifier computes loss to optimize the output and uses iterative learning process to ensemble results. We want to see if these features can become advantages on fitting our dataset. Among the model parameters, the loss function is set as deviance, and the learning rate is set to 0.1. The number of boosting stages is set to 100, as we hope that a large number of stages can lead to better model performance.

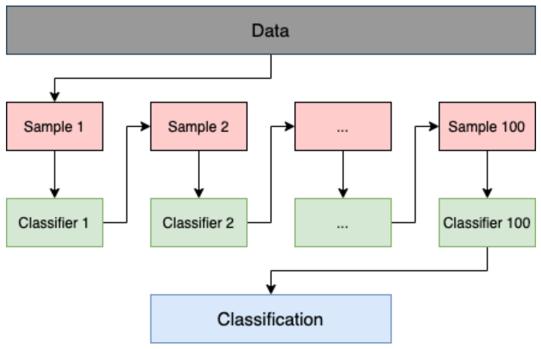

Fig. 4. Gradient Boosting Classifier Structure.

The other boosting model we use is the XGBoost. It is an optimized distributed library and implements algorithms under the Gradient Boosting framework, and it has some additional features. XGBoost is also called regularized form of Gradient Boosting Machine because it has in-built L1 and L2 regularization that prevents the model from overfitting. It also uses different tree pruning methods: it makes splits up to the maximum depth and start pruning the tree backwards and remove splits beyond which there is no positive gain. We use this model as a different implementation to fully understand the performance of Boosting algorithm on our dataset. For the XGB parameters we use "binary:logistic" for objective to perform a binary classification, and "auc" for eval_metric as the error function. The number of boosting round is also set to 100, and the train and test errors of each round are recorded and plotted. In the end, we use these errors to calculate the best iteration as the final predictions of the model.

C. GAN-based Training Data Generation

Generative Adversarial Network is a strategy we prepare for handling the unbalanced dataset. It is used to fit our dataset and generate some new data for each label class. The generated data has the same statistics as the original ones. Therefore, it is able to create some new data, and training classifiers with them has little side effects on the results. We also select GAN as a strategy because we realize the size of our dataset is relatively small. We notice a small dataset may lead to higher bias and less accurate data. Therefore we look for the GAN's functionality of generating data and hope that increasing the amount of data can also improve the accuracy of our results. A GAN is built up with three key components: a classifier, a discriminator, and a generator. Figure 5 shows the GAN structure. The classifier is the K-Means Clustering algorithm with two clusters. It classifies an input either as positive or negative for the coronavirus infection detection. The discriminator network consists of three dense layers and one sigmoid layer, and the generator network consists of four dense layers. The discriminator estimates whether a given input generated by the generator is synthetic or not. Its feedback helps the generator to generate synthetic data that cannot be detected as synthetic. The two models are trained together until the discriminator model is fooled most of the time, which means that the generator model can generate plausible inputs. Finally, the generated data is used as the input data to train all the previous classifiers again. The old and new results are then compared to see if there exist improvements on their performances and examine the effectiveness of the GAN.

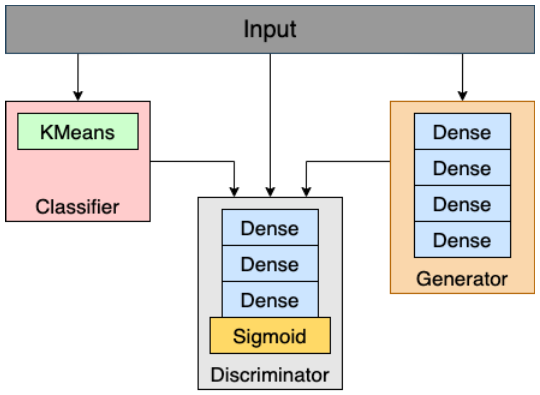

Fig. 5. Generative Adversarial Network Structure.

V. Performance Evaluation

A. Performance Metrics

The evaluation of all classifiers is mostly based on four metrics scores: Accuracy (Acc), Precision (Pre), Recall (Rec), and F1-score (F1). The accuracy evaluates the overall performance of the classifiers. Both precision and recall are important benchmarks, since we are looking for a predictor that accurately identifies both positive and negative cases. The F1-score as a weighted average of the precision and recall tells us about the relationship between precision and recall and indicate the advantages and drawbacks of the models. Therefore, we look for a predictor that is able to achieve high scores on all these metrics, and we select the classifier with best results to be our final predictor.

B. Correlation Analysis

Before training the classifiers, a correlation analysis is first performed on the prepared dataset. We decide to take this step after the data preprocessing because we want to involve both numerical and categorical columns, and therefore have to depend on the encoder outputs from the previous section. The goal of this analysis is to understand how strongly each feature column is correlated with the label column, and this information may be helpful for determining the weight of each feature in the training steps. We use the Pearson correlation coefficient (Corr) as the correlation score. It is the covariance of two variables divided by the product their standard deviations, and the result is a value between -1 and 1. To understand the result, -1 and 1 correspondingly indicate a strong negative and positive dependence between variables, while 0 indicates completely no dependence. To show the strength of correlations we only pay attention to the absolute value of each score. We compute the coefficient between the label column and every feature column and store the results. Figure 6 shows the absolute value of these coefficients as a bar chart, and the column name of each index are shown in Table I. The top ten features with highest correlation coefficients are: FirstRace, AdmittingDepartment, sbp, wght, dbp, heart_rate, Sex, cmp_bilirubin, cmp_ast, and pulse_ox. We hope this analysis may help us understand which features may have better influence on the infection detection results.

However, based on Figure 6 and the list, we observe that the the FirstRace feature has the highest correlation coefficient at around 0.22, and the AdmittingDepartment feature has the second highest correlation coefficient at around 0.09. Since the absolute values are plotted, higher values are able to indicate stronger correlation. Given the maximum value as 1, both 0.22 and 0.09 can only be regarded as a small value and therefore reflect very weak dependency between these two features and the label feature. For all the other columns, the coefficient values are even smaller. As a result, we observe from this correlation analysis that no strong dependence is found between the label column and any other feature column. We consider that the reason may be that the measure of Pearson correlation coefficient is able to only reflect a linear correlation of variables and may ignore many other types of relationship. Also, the skewed distribution shown in Figure 2 might also mislead the computation of correlation coefficients and result in the inaccurate display of the relationship between features.

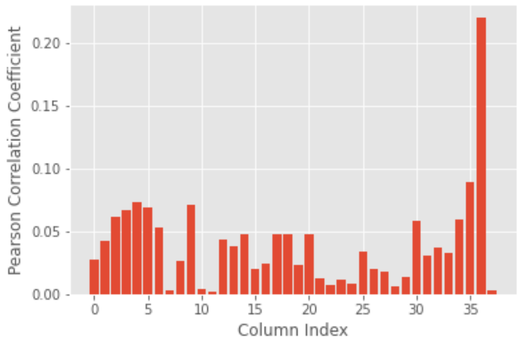

Fig. 6. Pearson Correlation Coefficient Results.

TABLE I
FEATURE (I.E., COLUMN) NAME.

Index	Feature Name		
0-2	Admitted	Age	Sex
3-5	heart_rate	sbp	dbp
6-8	pulse_ox	resp_rate	height
9-11	wght	cbc_wbc	cbc_hematocrit
12-14	cbc_hemoglobin	cbc_platelets	cbc_neutrophil_c
15-17	cbc_eosinophil_perc	cbc_lymphocyte_c	cbc_lymphocyte_perc
18-20	cbc_eosinophil_c	cbc_eosinophil_perc.1	cbc_monocyte_c
21-23	cbc_eosinophil_perc.2	cmp_sodium	cmp_potassium
24-26	cmp_creatinine	cmp_glucose	cmp_chloride
27-29	cmp_bicarbonate	cmp_bun	cmp_alt
30-32	cmp_ast	cmp_alkaline_phosphatase	cmp_total_protein
33-35	cmp_albumin	cmp_bilirubin	AdmittingDepartment
36-37	FirstRace	Ethnicity	

C. Performance of the Classifiers

1) ML on All Features: In this step, we use the prepared dataset to train and test all the classifiers planned in our approach. Through the testing process, we compare the predicted values with the actual ones and calculate the four metrics scores. Table II shows these scores of all the classifiers we train with the original dataset as the input. The XGBoost Classifier has the best accuracy at 94.58% among all the classifiers, but it has only moderate recall at 76.02% and low precision at 21.28%. This also makes its F1-score low, which is only 33.33%. Logistic Regression, Bagging Classifier, and Gradient Boosting Classifier also show the similar pattern in the results: all of them have accuracy results above 80%, but recall below 50% and precision below 25%. However, the results of Balanced Bagging Classifier is different and noticeable. It has very low accuracy at 8.13%, precision at 6.29%, and F1-score at 11.83%, but it is able to achieve 100.00% on the recall, which is much above our expectation. We consider that this may be attributed to the additional sample balancing steps during the training process, and these extra steps modify the data samples and therefore lead to different results. In general, to be considered as achieving good performances, a classifier needs to have high scores for all four metrics. None of them is able to meet this expectation, so all the classifiers have poor overall performances when using the original dataset as the input.

TABLE II
CLASSIFIER RESULTS.

Classifier	Accuracy	Precision	Recall	F1-score
Logistic Regression	87.40%	23.16%	45.05%	30.60%
Bagging Classifier	86.92%	22.58%	46.15%	30.32%
Balanced Bagging	8.13%	6.29%	100.00%	11.83%
Gradient Boosting	84.42%	14.72%	31.87%	20.14%
XGBoost Classifier	94.58%	21.28%	76.92%	33.33%

2) ML on Chi-square Test based Selected Features: Our feature selection step begins with the Chi-square test (Chi) method. We first compute the Chi-square test score for all the features in the dataset. The results are separately shown in two parts: Figure 7 shows the results of column index from 0 to 33, and Table III shows the results of column index from 34 to 37. We observe the results on column index 34, 35, and 36 contain much larger values than the others, and plotting all of them together scales the figure and makes the difference between small values indistinguishable. The column names of each index are consistent with the correlation coefficient results and can be found in Table I.

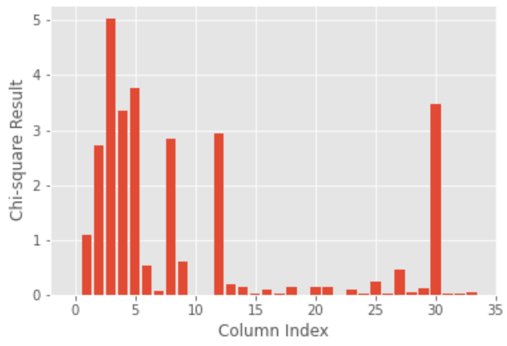

Fig. 7. Chi-square Test Results 0-33.

TABLE III
CHI-SQUARE TEST RESULTS 34-37.

Index	Result
34	280.13305294349914
35	91.54388392569649
36	34.15832619081337
37	0.012657909041169185

Next, by sorting the result we are able to obtain the best 10, 20, and 30 features. There exists different pattern in Figure 7 and Figure 6, and we compare these top features with the identified features of the Correlation Coefficient to determine how effective our Chi-square method is in identifying the key features. Table IV displays the comparison results. These two methods have 8 common features in the top 10 features, 14 common features in the top 20 features, and 18 common features in the top 30. We notice the Chi-square test is most effective in identifying top 10 features. As the number of features increases, its performance gets worse. The top 10 features are: cmp_bilirubin, AdmittingDepartment, FirstRace, heart_rate, dbp, cmp_ast, sbp, cbc_hemoglobin, height, and Sex.

Next, we train the previous Logistic Regression model using the top 10, 20, and 30 features selected by our Chi-square test. We believe the Logistic Regression as the most basic classification model is able to best reflect the effectiveness of our Chi-square test feature selection method. Table V shows the metrics score results. The best performance is generated when only the top 10 features are used as the input. Among all the four cases, top 10 features is able to achieve highest accuracy, precision, and F1-score at 88.55%, 25.00%, and 31.58%. The recall is 42.86%, which is not the highest one but still close enough to others' recall results. This accords with our previous observation from Table IV. Also, the results from using the top 10 features are slightly higher than the values shown in Table II. The accuracy increases 1.55%, the precision increases 1.84%, and the F1-score increases 0.98%. Even though the recall decreases 2.19%, we can still consider that the Chi-square test method can help improve the accuracy performance of the classifier slightly and also reduce the ML computing overhead by reducing the input features.

TABLE IV
CORRELATION AND CHI-SQUARE COMPARISON.

Features	**Method**	**Column Index**	**Common**
Top 0-9	Corr	**36**, **35**, **4**, 9, **5**, **3**, **2**, **34**, **30**, 6	8
	Chi	**34**, **35**, **36**, **3**, **5**, **30**, **4**, 12, 8, **2**	
Top 10-19	Corr	**14**, **18**, **20**, 17, 12, **1**, **13**, 32, **25**, 33	6
	Chi	**1**, 9, 6, 27, **25**, **13**, 21, **14**, **20**, **18**	
Top 20-29	Corr	**31**, 0, 8, **16**, 19, **15**, 26, 27, **29**, 21	4
	Chi	**29**, 23, **16**, 7, 33, 28, **31**, 32, **15**, 24	

TABLE V
LR RESULTS WITH CHI-SQUARE TEST.

Features	**Acc**	**Pre**	**Rec**	**F1**
All	87.40%	23.16%	45.05%	30.60%
Top 10	88.55%	25.00%	42.86%	31.58%
Top 20	87.26%	22.60%	43.96%	29.85%
Top 30	87.20%	22.78%	45.05%	30.26%

3) ML on Knowledge-based Selected Features: We also respectively use the 12, 10, 9, 8, and 7 features that are manually selected based on knowledge as explained in Section IV-A as the input of the classifier for prediction. We pick a subset that contains exactly these features from the dataset, and we precisely follow the designed procedure to remove some features from this subset of data. We then train all the previous classifiers on these subsets, and the results of all metrics scores are shown in Table VI. Among the five different number of features, Logistic Regression has the best accuracy at 89.02% when using 7 features as its input. However, using of 9 features makes it achieve the best precision at 29.11%, recall at 46.00%, and F1-score at 35.66%. The accuracy in this case is 88.75%, only 0.27% lower than the accuracy achieved with 7 features. Therefore we consider that Logistic Regression has the best overall performance with 9 features. The results of Logistic Regression are also comparable with the best results, which are the results of top 10 features in Table V. We observe slight improvement of the performances on all metrics scores. There are 0.20% increments in accuracy, 4.11% in precision, 3.14% in recall, and 4.08% in F1-score. These improvements can show that our knowledge-based selected features are better for the feature selection than the Chi-square test method. We also compare the results between different classifiers and notice that the XGBoost Classifier again has better accuracy than the others. It has the best results with 12 features and achieves 93.90% on accuracy. It also has 61.11% on recall, which is relatively higher than other models. However, it only has 11.70% on the precision and 19.64% on the F1-score. These values are lower than those of Logistic Regression and Bagging Classifier. The Balanced Bagging Classifier has noticeable and unchanged results for all different number of features. This probably shows that changing the number of features does not influence the sample balancing within the model. The Bagging Classifier has the best accuracy at 89.91% and precision at 31.01% on 8 features, and the best recall at 46.00% and F1-score at 35.94% on 9 features. For the Gradient Boosting Classifier, it has the best accuracy at 77.10% on 9

TABLE VI
RESULTS OF ML PREDICTION USING KEY FEATURES.

Classifier	**Scores**	**Number of features remaining**				
		12	**10**	**9**	**8**	**7**
LR	Acc	88.62%	88.62%	88.75%	88.89%	89.02%
	Pre	28.75%	28.75%	29.11%	28.08%	28.47%
	Rec	46.00%	46.00%	46.00%	41.00%	41.00%
	F1	35.38%	35.38%	35.66%	33.33%	33.61%
BC	Acc	88.89%	88.82%	88.89%	89.91%	89.84%
	Pre	29.49%	29.30%	29.49%	31.01%	30.77%
	Rec	46.00%	46.00%	46.00%	40.00%	40.00%
	F1	35.94%	35.80%	35.94%	34.93%	34.78%
BBC	Acc	7.93%	7.93%	7.93%	7.93%	7.93%
	Pre	6.85%	6.85%	6.85%	6.85%	6.85%
	Rec	100.00%	100.00%	100.00%	100.00%	100.00%
	F1	12.83%	12.83%	12.83%	12.83%	12.83%
GBC	Acc	73.04%	76.63%	77.10%	72.63%	72.15%
	Pre	6.69%	7.61%	7.19%	5.29%	5.19%
	Rec	23.00%	22.00%	20.00%	18.00%	18.00%
	F1	10.36%	11.31%	10.58%	8.18%	8.05%
XGB	Acc	93.90%	93.36%	93.43%	93.56%	93.43%
	Pre	11.70%	2.13%	4.26%	2.13%	2.13%
	Rec	61.11%	25.00%	36.36%	40.00%	28.57%
	F1	19.64%	3.92%	7.62%	4.04%	3.96%

features, the highest recall at 23.00% on 12 features, and the highest precision and F1-score at 7.61% and 11.31% on 10 features. All models except the Balanced Bagging Classifier still have their results in the pattern of high accuracy and low other scores. This shows that even though as a better feature selection method than the Chi-square test, our knowledge-based selection method is still not able to sufficiently improve the results of any classifier to an expected level.

Comparing Table VI with Table II, we see that the accuracy results of Logistic Regression and Bagging Classifier with 12 input features are higher than those with all input features, and the accuracy results of other models with all input features are lower than those with all input features. When the 12 features instead of all features are used as the input, the other metrics of Logistic Regression and Bagging Classifier also increase, the other metrics of Gradient Boosting Classifier and XGBoost Classifier also decrease, but for Balanced Bagging Classifier some metrics increase and some decrease. In addition, when the number of input features decreases, the different metrics for a given model generally decrease very slightly, and some metrics do not necessarily decrease or even increase. Therefore, the absence of some features may not affect the infection detection accuracy, and the influence degree depends on the ML model that is used.

D. GAN-assisted ML based Prediction

Based on the original dataset, we create a balanced dataset using the GAN introduced above and use this new dataset to perform training and testing on all classifiers. The most important goal of this step is to identify how the skewed data influences the outcomes of classifiers, and how the GAN generated new balanced dataset improves the performance. To train our GAN, we pass in all the 7380 rows of data from the original dataset as its input. The GAN in return creates two new dataset: one for training and the other for testing. Both of them contain 7380 rows of data. We first evaluate the distribution of these generated dataset, and Figure 8 shows the value count as a bar chart. Compared to Figure 2, both of the newly generated dataset are balanced and contain exactly 3690 (50%) positive cases.

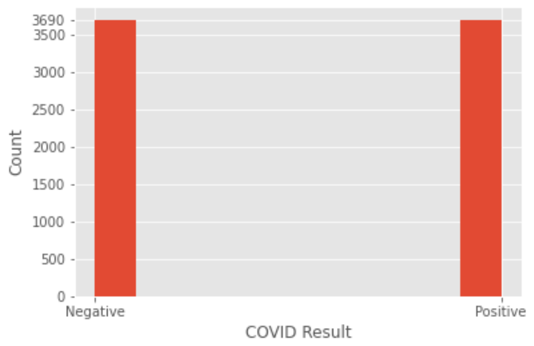

Fig. 8. Generated Data Distribution.

All classifiers are then separately trained and tested again on the new train and test sets, and metrics scores are displayed in Table VII. Compared to Table II, we observe great improvements on all the metrics scores of every classifier. Inside this table, the best classifier is the Gradient Boosting Classifier, with 100.00% on all scores. The XGBoost Classifier also has high performances and achieves above 99% on all scores. Therefore, we can know that Boosting algorithm generally has better performances on the GAN generated dataset. For the other models, Logistic Regression and Bagging Classifier maintain the accuracy above 85% and have huge improvements on the other three scores. Logistic Regression has 82.44% on precision, 98.48% on recall, and 89.75% on F1-score. Bagging Classifier has 82.82% on precision, 98.40% on recall, and 89.94% on F1-score. For the Balanced Bagging Classifier, though the recall slightly reduces from 100.00% to 98.48%, the other scores are improved from poor to high: the accuracy achieves 89.20%, the precision achieves 83.06%, and the F1-score achieves 90.12%. These results reflect that the balanced dataset generated by the GAN is able to improve the performances of all the classifiers to an expected level.

Also, in order to further understand the effectiveness of GAN and feature selection on improving performances, we perform the knowledge-based feature selection method, which has been shown to be better than the Chi-square test, on the GAN generated dataset. Classifiers are trained again and the results are shown in Table VIII. Compared to results in Table VI, we also observe huge improvements on all classifiers. With the original dataset, Balanced Bagging Classifier has all but recall results below 15%, and the other models have all but accuracy results below 65%. In contrast, with the generated dataset, Balanced Bagging Classifier has great improvements on accuracy, precision, and F1-score, while the other models have improvements on precision, recall, and F1-score. Every model is able to achieve at least 75% on all metrics, and the Gradient Boosting Classifier still have the best performances. This indicates that GAN contributes significantly to the performance improvements. For the number of features we observe different patterns from the previous ones with original dataset. When using 12 features instead of all as the input, Logistic

TABLE VII
RESULTS OF GAN-ASSISTED ML PREDICTION.

Classifier	Accuracy	Precision	Recall	F1-score
LR	88.75%	82.44%	98.48%	89.75%
BC	89.00%	82.82%	98.40%	89.94%
BBC	89.20%	83.06%	98.48%	90.12%
GBC	100.00%	100.00%	100.00%	100.00%
XGB	99.96%	99.92%	100.00%	99.96%

TABLE VIII
RESULTS OF GAN-ASSISTED ML PREDICTION USING KEY FEATURES.

Classifier	Scores	Number of features remaining				
		12	10	9	8	7
LR	Acc	85.27%	85.05%	81.34%	87.25%	83.43%
	Pre	78.47%	78.41%	75.09%	81.77%	76.96%
	Rec	97.21%	96.75%	93.79%	95.88%	95.42%
	F1	86.84%	86.62%	83.41%	88.26%	85.20%
BC	Acc	85.16%	84.99%	81.23%	87.21%	83.37%
	Pre	78.40%	78.39%	75.07%	81.77%	76.97%
	Rec	97.07%	96.61%	93.52%	95.77%	95.26%
	F1	86.74%	86.55%	83.29%	88.22%	85.14%
BBC	Acc	85.16%	85.07%	81.29%	87.24%	83.41%
	Pre	78.35%	78.41%	75.03%	81.76%	76.97%
	Rec	97.18%	96.78%	93.79%	95.85%	95.37%
	F1	86.75%	86.63%	83.37%	88.25%	85.19%
GBC	Acc	100.00%	99.96%	99.99%	99.95%	100.00%
	Pre	100.00%	100.00%	99.97%	100.00%	100.00%
	Rec	100.00%	99.92%	100.00%	99.89%	100.00%
	F1	100.00%	99.96%	99.99%	99.95%	100.00%
XGB	Acc	99.99%	99.93%	99.95%	99.96%	99.92%
	Pre	99.97%	99.97%	99.97%	100.00%	99.84%
	Rec	100.00%	99.89%	99.92%	99.92%	100.00%
	F1	99.99%	99.93%	99.95%	99.96%	99.92%

Regression, Bagging Classifier, and Balanced Bagging Classifier have lower accuracy results this time. As the number of features decrease to 9, the accuracy results decrease as well. However, when using 8 features, these three classifiers have accuracy results higher than those with 12 features but lower with those with all, and then the results decrease again when the number becomes 7. For Gradient Boosting Classifier and XGBoost Classifier, in all cases they have every metrics above 99%, and changing number of features has little effect on their performances. For example, when 8 features instead of 9 are used, Gradient Boosting Classifier has accuracy results decreased by 0.04% and XGBoost Classifier has accuracy increased by 0.01%, while the other models have accuracy increased by around 6%. The difference between models indicates that the influence degree mostly depends on the ML model rather than the number of features, and the best ML predictor still can provide relatively high performance when some features are not available.

E. Summary

From the experiment we obtain five sets of results: one on the original dataset in Table II, two with the feature selection methods in Figure 7, Table III, and Table VI, and two on the GAN generated dataset in Table VII and Table VIII. By analyzing and comparing each of these results, we notice that the GAN generated dataset yield great performances on all classifiers, and the original dataset always lead to some poor metric scores in the result. Also, no matter which dataset is used, the feature selection process does not lead to notable improvement in the performances. Therefore, we are able to confirm that the poor results on the original prepared data are mainly attribute to the skewed distribution of the dataset, but not the interference of irrelevant features. For the results on generated data, all classifiers achieve above 80% on all evaluation metrics. The classifier with best performance is the Gradient Boosting Classifier. It achieves 100.00% on all metrics and is much better than an expected performance.

We summarize our findings from our experiments below:

- For insufficient or unbalanced training data, GAN is able to generate balanced dataset based on an extremely skewed dataset. The comparison between Figure 2 and Figure 8 is able to reflect its effectiveness.
- With the GAN generated dataset, Boosting algorithm in general has better performances than the Bootstrap Aggregating algorithm and Logistic Regression, according to the results in Table VII and Table VIII.
- Our Knowledge-based feature selection is a better method than the Chi-square test. This is reflected by comparison between the results of Logistic Regression in Table V and Table VI.
- It is necessary to use selected features instead of all from the dataset to train the classifiers in order to save the training overhead. Our feature selection test results shown in Table VI and Table VIII reveals that removing unimportant features only slightly decreases the infection detection accuracy and does not significantly affect the results.
- The quality of results depends more on the ML model but not the features selected, according to Table VI and Table VIII.

VI. CONCLUSION

In this study, we developed predictors of COVID-19 infection using different machine learning models. We test the performances of multiple classifiers and show that compared with other classifiers, the Gradient Boosting Classifier that implements the Boosting technique is able to achieve the best performance on all the metrics scores. We also apply two feature selection methods on our dataset and find that they are only able to slightly improve the model results. The usage of Generative Adversarial Network helps us recreate a new balanced dataset based on the original one. The high accuracy performance of trained classifiers based on these new dataset clearly indicates that the previous poor results are mainly due to the skewed data distribution. We believe our model can provide useful guidance on the study of using ML techniques for COVID-19 infection detection for patients and non-patients to some degrees. In our future work, we will keep testing and improving the accuracy of our predictor with more incoming patients' data and explore whether deep learning models will achieve more accurate and reliable prediction and study their extra overhead. Once we believe our predictor is sufficiently reliable, we may deploy it to an existed system for further testing and evaluation. We also realize that the skewed data distribution will still be a potential issue in the future work, and we understand it is necessary to look for solutions to allow the classifiers to perform well on both balanced and skewed dataset.

ACKNOWLEDGEMENTS

This research was supported in part by U.S. NSF grants NSF-1827674, CCF-1822965, FHWA grant 693JJ31950016, Microsoft Research Faculty Fellowship 8300751, and AWS ML Research Award. We thank Mr. Tanmoy Sen for his help during the work of this paper.

REFERENCES

[1] J. J. Amon, "Covid-19 and detention: Respecting human rights." *Health and Human Rights,*, vol. 22, no. 1, pp. 367–370, 2020.

[2] Worldometer, "Covid-19 coronavirus pandemic," December 2020. [Online]. Available: https://www.worldometers.info/coronavirus/\#countries

[3] Centers for Disease Control and Prevention, "Symptoms of coronavirus," December 2020. [Online]. Available: https://www.cdc.gov/coronavirus/2019-ncov/symptoms-testing/symptoms.html

[4] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013, vol. 398.

[5] W. Fan and K. Zhang, *Bagging: Encyclopedia of Database Systems*. Springer US, 2009.

[6] Imbalanced-learn API, "Balanced bagging classifier." [Online]. Available: https://imbalanced-learn.org/stable/generated/imblearn.ensemble.BalancedBaggingClassifier.html

[7] J. H. Friedman, "Stochastic gradient boosting," *Elsevier Science Publishers B. V.*, vol. 38, no. 4, 2002.

[8] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016. [Online]. Available: http://arxiv.org/abs/1603.02754

[9] Scikit-learn Chi2, "sklearn.feature_selection.chi2," 2011. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html\#sklearn.feature_selection.chi2

[10] C. Nash, "Create data from random noise with generative adversarial networks," 2017. [Online]. Available: https://www.toptal.com/machine-learning/generative-adversarial-networks

[11] Yan, Li *et al.*, "A machine learning-based model for survival prediction in patients with severe covid-19 infection," *medRxiv*, 2020. [Online]. Available: https://www.medrxiv.org/content/early/2020/03/17/2020.02.27.20028027

[12] Wu, Jiangpeng *et al.*, "Rapid and accurate identification of covid-19 infection through machine learning based on clinical available blood test results," *medRxiv*, 2020. [Online]. Available: https://www.medrxiv.org/content/early/2020/04/06/2020.04.02.20051136

[13] D. Brinati, A. Campagner, D. Ferrari, M. Locatelli, G. Banfi, and F. Cabitza, "Detection of covid-19 infection from routine blood exams with machine learning: A feasibility study." *Journal of Medical Systems*, vol. 44, no. 8, p. 135, 2020. [Online]. Available: https://doi.org/10.1007/s10916-020-01597-4

[14] Yao, Haochen *et al.*, "Severity detection for the coronavirus disease 2019 (covid-19) patients using a machine learning model based on the blood and urine tests," *Frontiers in Cell and Developmental Biology*, vol. 8, p. 683, 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/fcell.2020.00683

[15] Assaf, Dan *et al.*, "Utilization of machine-learning models to accurately predict the risk for critical covid-19," *Internal and Emergency Medicine*, vol. 15, no. 8, pp. 1435–1443, 2020. [Online]. Available: https://doi.org/10.1007/s11739-020-02475-0

[16] Heldt, Frank S. *et al.*, "Early risk assessment for covid-19 patients from emergency department data using machine learning," *Scientific Reports*, vol. 11, no. 1, p. 4200, 2021. [Online]. Available: https://doi.org/10.1038/s41598-021-83784-y

[17] Schöning, Verena *et al.*, "Development and validation of a prognostic covid-19 severity assessment (cosa) score and machine learning models for patient triage at a tertiary hospital," *Journal of Translational Medicine*, vol. 19, no. 1, p. 56, 2021. [Online]. Available: https://doi.org/10.1186/s12967-021-02720-w

[18] A. F. de Moraes Batista, J. L. Miraglia, T. H. Rizzi Donato, and A. D. Porto Chiavegatto Filho, "Covid-19 diagnosis prediction in emergency care patients: a machine learning approach," *medRxiv*, 2020.

[19] Cheng, Fu-Yuan *et al.*, "Using machine learning to predict icu transfer in hospitalized covid-19 patients," *Journal of Clinical Medicine*, vol. 9, no. 6, 2020. [Online]. Available: https://www.mdpi.com/2077-0383/9/6/1668

[20] M. A. Awal, M. Masud, M. S. Hossain, A. A.-M. Bulbul, S. M. H. Mahmud, and A. K. Bairagi, "A novel bayesian optimization-based machine learning framework for covid-19 detection from inpatient facility data," *IEEE Access*, vol. 9, pp. 10 263–10 281, 2021.

[21] S. Vaid, C. Cakan, and M. Bhandari, "Using machine learning to estimate unobserved covid-19 infections in north america," *The Journal of bone and joint surgery. American volume*, vol. 102, no. 13, p. 70, 2020. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/32618918

[22] The Lancet Digital Health, "Harnessing wearable device data to improve state-level real-time surveillance of influenza-like illness in the usa: A population-based study," 2020. [Online]. Available: https://www.thelancet.com/journals/landig/article/PIIS2589-7500(19)30222-5/fulltext

[23] ClinicalTrials.gov, "The detect(digital engagement and tracking for early control, and treatment) study," 2020. [Online]. Available: https://clinicaltrials.gov/ct2/show/NCT04336020

[24] Healthcare IT News, "Scripps, stanford working with fibit to assess wearables' covid-19 tracking abilities. healthcare it news," 2020. [Online]. Available: https://www.healthcareitnews.com/news/scripps-stanford-working-fibit-assess-wearables-covid-19-tracking-abilities

[25] D. Etherington, "Researchers to study if startup's wrist-worn wearable can detect early covid-19 respiratory issues," 2020. [Online]. Available: https://social.techcrunch.com/2020/04/01/researchers-to-study-if-startups-wrist-worn-wearable-can-detect-early-covid-19-respiratory-issues/

[26] Duke University, "A duke university study." [Online]. Available: https://covidentify.org/

[27] Duke Pratt School of Engineering, "Covidentify pits smartphones and wearable tech against the coronavirus," 2020. [Online]. Available: https://pratt.duke.edu/about/news/covidentify-pits-smartphones-and-wearable-tech-against-coronavirus

[28] Scikit-learn Simple Imputer, "sklearn.impute.simpleimputer," 2011. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html\#sklearn.impute.SimpleImputer

[29] Scikit-learn Normalizer, "sklearn.preprocessing.normalizer," 2011. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normalizer.html\#sklearn.preprocessing.Normalizer

[30] Scikit-learn Standard Scaler, "sklearn.preprocessing.standardscaler," 2011. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html\#sklearn.preprocessing.StandardScaler

[31] Scikit-learn Ordinal Encoder, "sklearn.preprocessing.ordinalencoder," 2011. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html\#sklearn.preprocessing.OrdinalEncoder

[32] Scikit-learn Label Encoder, "sklearn.preprocessing.labelencoder," 2011. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html\#sklearn.preprocessing.LabelEncoder

[33] Li, Qun *et al.*, "Early transmission dynamics in wuhan, china, of novel coronavirus–infected pneumonia," *New England Journal of Medicine*, vol. 382, no. 13, pp. 1199–1207, 2020, pMID: 31995857. [Online]. Available: https://doi.org/10.1056/NEJMoa2001316

[34] Seshadri, Dhruv R. *et al.*, "Wearable sensors for covid-19: A call to action to harness our digital infrastructure for remote patient monitoring and virtual assessments," *Frontiers in Digital Health*, vol. 2, p. 8, 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/fdgth.2020.00008

[35] Pedregosa, F. *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[36] Lars Buitinck *et al.*, "API design for machine learning software: experiences from the scikit-learn project," in *Proc. of ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[37] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html

[38] Imbalanced-learn, "Randomundersampler," 2020. [Online]. Available: https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html#imblearn.under_sampling.RandomUnderSampler