# Supervised Algorithms for the Detection of COVID-19 from Chest Scan Images

Amr Alfayoumy

*Electronic and Computer Engineering Department*

*Nile University*

*Giza, Egypt*

a.alfayoumy@nu.edu.eg

*Abstract*— **In this paper, a supervised machine learning algorithm is designed and proposed for two classification models using publicly available datasets. In the first model, the CT model is able to determine whether a given chest CT image of a patient is infected with COVID-19 or not with 98.2% average accuracy. As for the second model, the X-ray model is able to detect COVID-19 in a patient's X-ray image with 99.6% average accuracy. The proposed model uses image feature extraction methods and requires relatively low computational power to train and test the model compared to the currently available methods in the literature. The model may also lead to the early detection of COVID-19 and limit its spreading in the patients' respiratory systems. A total of 2,471 CT and 1,227 X-ray scan images are used to train and test the models. This paper aims to provide an efficient low-cost alternative to PCR tests to detect COVID-19, particularly in rural areas where patients cannot afford the tests and countries where the necessary resources to conduct widespread testing are not available.**

***Keywords*—- COVID-19, supervised learning, coronavirus detection**

## I. Introduction

The coronavirus pandemic, also known as the COVID-19 pandemic, is an ongoing pandemic of coronavirus disease (COVID-19) caused by the transmission of severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), which was first identified in December 2019 in Wuhan, China [1]. The COVID-19 outbreak was declared a Public Health Emergency of International Concern in January 2020, and a pandemic in March 2020 [2]. The novel virus is responsible for the general destruction of respiratory systems and severe respiratory symptoms which are often associated with highly Intensive Care Unit (ICU) admissions and death. As of January 4, 2021, the total number of confirmed COVID-19 cases worldwide was 85,362,013 and the number of deaths due to COVID-19 disease was 1,847,146 according to the Johns Hopkins Coronavirus Research Center [3].

Research on the detection and treatment of the COVID-19 has gained great momentum. The main reasons the virus is deemed dangerous are: it's highly contagious, it can be contagious before the patient develops symptoms, it can lead to death as a result of acute respiratory failure and lung pneumonia, and there is no effective anti-SARS-CoV-2 drug for the treatment of patients with COVID-19 [4]. While a few vaccinations have been developed and approved, projections from the Duke Global Health Innovation Center predict that people in low-income countries will have to wait for vaccinations until 2023 or 2024 [5]. Meanwhile, rich nations are hoarding the most promising COVID-19 vaccines. Although rich nations represent just 14% of the world's population, they have acquired 53% of the vaccines so far [6], and it is estimated that 9 out of 10 people in poor countries are unlikely to receive a COVID-19 vaccine in 2021 [7].

In the past decade, computer vision and image processing have greatly evolved. It has been one of the leading technologies that has the greatest healthcare potential in healthcare. Because of its significant role in healthcare applications, medical imaging has gained a lot of attention. Computer vision algorithms work in a manner similar to a human eye to find patterns and irregularities in images to achieve a diagnosis One of the key characteristics of Artificial Intelligence and a constant necessity in healthcare is to evaluate complicated circumstances, make predictions, and determine patterns. Well-executed AI algorithms, by pointing out contradictions and enhancing care, can to save lives [8].

The CT model is able to determine whether a given chest CT image of a patient is infected with COVID-19 or not with 98.2% average accuracy. As for the X-ray model, the model is able to detect COVID-19 in a patient's X-ray image with 99.6% average accuracy. The proposed, which is a massive improvement to available methods in the literature, as well as the relatively low computational power needed to train and test the model compared to the currently available methods. The model is also able to start detecting COVID-19 in lung X-ray scan images as early as 4 days prior to infection, which may lead to the early detection of COVID-19 and limit its spreading in the patients' respiratory systems.

The goal of this study is to use feature extraction methods and supervised learning algorithms to structure a novel efficient model that requires massively less processing power than most of the models that currently exist in the literature, which rely heavily on neural networks and the use of graphics processing units (GPU) while producing results that are comparable to state-of-the-art methods. The paper is organized as follows: the following section presents the supervised learning models used in the proposed method. Then, the Results and Discussion section offers an analysis of the experimental results. The final section covers the conclusions.

## II. Methods

The early diagnosis of COVID-19 leads limits its spreading in the patients' respiratory systems and increases the recovery rates. Medical imaging modalities, especially the Computed Tomography (CT) and X-ray scans are able to detect the changes in the lungs caused by the COVID-19 virus. This research aims to employ different Artificial Intelligence (AI), specifically supervised machine learning algorithms, to propose a system for the early detection of COVID-19 using chest CT and X-ray scan images. These images are classified using different AI algorithms, then their performance is evaluated. Features are first extracted from the images, then passed to supervised learning algorithms that 'learns' patterns from the given labeled data to accurately identify the COVID-19 scan images from the non-coronavirus ones. The algorithms used are support vector machines (SVM), K-nearest neighbor (KNN), and Random Forest Classifier. Voting Classifier is then used to ensemble the separate models into one, more accurate model. The features extracted are: raw pixel intensities, 3D color histograms, Haralick Texture features, and Threshold Adjacency Statistics. The performance of each algorithm using each feature is calculated to determine the best-performing algorithm, then a new model is structured using the best-performing features and then evaluated against novel scan images input.

### A. *Dataset*

Since the COVID-19 virus has only appeared very recently, finding an appropriate dataset is a challenging task. This paper uses two publicly available datasets. The first dataset is collected from real patients in hospitals in Sao Paulo, Brazil. The dataset contains 1,252 CT scans for patients that are COVID-19 positive and 1,230 CT scans for patients who are not infected by COVID-19, a total of 2,482 CT scans [9]. From this dataset 1,247 images are used for positive COVID-19 cases and 1,224 for negative cases. The size of the CT dataset used in this paper is 235 MB. The second dataset is collected by a team of researchers from Qatar University, and the University of Dhaka, Bangladesh along with their collaborators from Pakistan and Malaysia. The dataset contains 1,143 COVID-19 positive X-ray images, 1,341 X-ray images for 'normal' lungs, and 1,345 X-ray images for viral pneumonia that is not caused by SARS-CoV-2 [10]. From this dataset, only 1,227 images are used: 600 images for COVID-19 positive cases and 627 images for negative cases that include images of both 'normal' lungs and 'viral pneumonia'. The final size of the X-ray dataset used in this paper is 276 MB. A sample of each dataset is shown in Fig. 1.

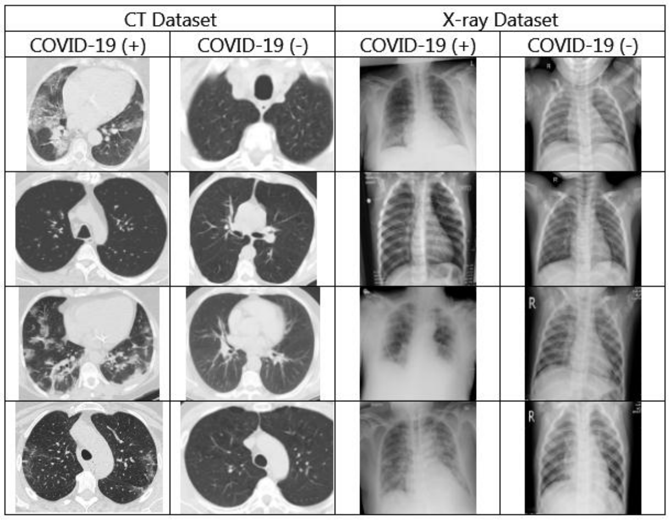


Fig. 1 A sample of each of the CT and X-ray scan images datasets. Each sample contains 4 chest scan images of COVID-19 positive patients and 4 chest scan images of COVID-19 negative patients.

The two datasets used in this study are publicly available, and their corresponding websites are referenced in this paper. Figure 3 shows a sample from the datasets of the chest X-ray images and chest CT scan images that tested positive for the COVID-19 virus and others that tested negative for the COVID-19 virus.

## B. Feature Extraction

### 1) Raw Pixel Intensities

Image Feature Vector is a numerical quantification of an image used to identify its characteristics. The simplest feature vector is raw pixel intensities. A raw pixel intensities feature vector is a list of numbers corresponding to the raw BGR pixel intensities of an image [11]. First, to maintain a reasonable memory use, the scan images are resized to a fixed size, then the resulting 3-dimensional array is flattened into a 1-dimensional list of raw pixel intensities. resize() function of OpenCV libraries is used to resize the image into (width, height) of (60, 60) pixels and to extract the pixel values, as shown in Fig. 2.

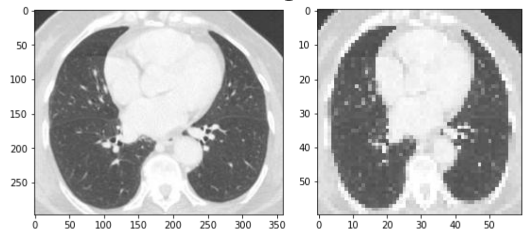


Fig. 2 A CT scan image plotted before performing OpenCV's resize() function (left) and after performing the resize() function (right).

The result is a NumPy array of shape (60, 60, 3) as this scans the image horizontally from left to right starting at the top-left corner. The result is a NumPy array with each pixel value as a set of 3 values in the order of BGR (blue, green, red), and each pixel value is an integer that ranges from 0 (black) to 255 (white).

It is now needed to flatten the array using NumPy's flatten method. The Blue, Green, Red components of the image have been flattened into a single list (rather than a multi-dimensional array). This yields a copy of the array collapsed into one dimension. The result is a NumPy array of the shape (10800,) since 60×60×3 is 10800. To reduce the memory size needed, the scan image is first converted to grayscale which uses 1 channel instead of 3 channels. Each pixel value now is 1 value instead of a set of 3 values, and the value ranges from 0 (black) to 255 (white). The result is a NumPy array of 60×60×1 shape (3600,) rather than 60×60×3 (10800,).

### 2) HSV Color Histograms

HSV is a color model that offers another representation of the RGB. HSV, along with HSL, were designed by computer graphics researchers to provide computer vision the ability to perceive color-making attributes more similarly to human vision [12]. The HSV (Hue-Saturation-Value) model specifies a color by three numbers: H (Hue), S (Saturation), and V (Value). The hue corresponds to an angle from 0 to 360 degrees, the saturation measures the departure of a hue from white or gray and has a range of [0,1], and the value measures the departure of a hue from black, the color of zero energy, and has a range of [0,1] [13].

Unlike RGB colorspace, HSV separates the image intensity (luma), from the color information (chroma). This is very useful in performing histogram equalization of a color image, as we want to do that only on the intensity component, leaving the color components untouched. RGB and HSV color spaces are visualized in Fig. 3.

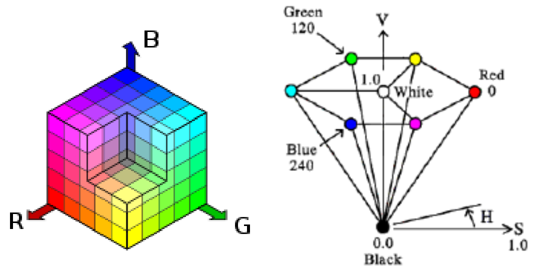


Fig. 3 RBG color space (left) vs. HSV hexacone (right).

A histogram is another way of understanding the image through a plot or a graph of an image's intensity distribution. It is a plot with pixel values (ranging from 0 to 255) on the X-axis and the corresponding number of pixels in the image on the Y-axis. The histogram of an image provides insight into the contrast, brightness, intensity distribution, and other features of an image, and almost all image processing tools today provide features on histograms.

The following OpenCV function [14] is used to build the HSV histogram in this paper:

cv2.calcHist(images, channels, mask, histSize, ranges)

images: This will be the image for which the function will be computing the histogram. It will be passed to this function in the form of a list.

channels: Channels are a list of indexes. This specifies the index of the channel for which the histogram will be computed. Since the HSV histogram is being computed for the HSV image, so the Hue, Saturation, and Value channels list would be [0,1,2].

mask: Since we are not applying a mask and simply computing the histogram for the entire image then we will specify "None" in that case.

histSize: This specifies the bin size that will be used to compute the histogram. This is a list with each value in the list corresponding to each channel. In this paper, [8,8,8] corresponds to 8 bins per channel.

ranges: This is to specify the range of the pixel values. Normally, we have pixel ranges within [0,256] if we are dealing with the BGR scale, however, this is different here as we are using the HSV scale. OpenCV uses the following ranges to represent each of the parameters in the HSV spectrum: Hue: [0, 180], Saturation: [0, 256], Value: [0, 256].

To calculate the 3D color histogram for the HSV image, first the scan image is converted the BGR colorspace to HSV through cv2.cvtColor(image, cv2.COLOR_BGR2HSV). Then, the histogram is calculated for each channel separately through the following: hist_h = cv2.calcHist([image], [0], None, [180], [0, 180]) for Hue channel; hist_s = cv2.calcHist([image], [1], None, [256], [0, 256]) for Saturation channel; hist_v = cv2.calcHist([image], [2], None, [256], [0, 256]) for Value channel. Finally, the 3 arrays are horizontally merged into one via NumPy's concatenate method. A sample of HSV color histograms calculated for two CT scan images is shown in Fig. 4.

### 3) *Haralick Textures*

'Texture' is the consistency of patterns and colors in an image. To classify images based on texture, we look for consistent patterns and colors in an image. Haralick Texture [15] is used as a way to quantify an image based on texture. 'Haralick Texture' features can be computed using the fundamental concept of the Gray Level Co-occurrence Matrix or GLCM.

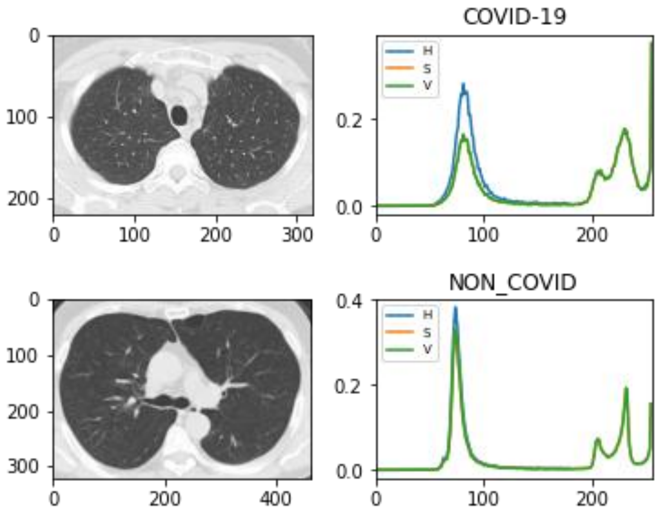


Fig. 4 HSV histograms plotted for a CT scan image of a COVID-19 positive case (above) and a CT scan image of a COVID-19 negative case (below). The dominant lines are for Hue in blue and Value in green.

Gray Level Cooccurrence Matrix (GLCM) uses the "adjacency" concept in images to scan the image for pairs of adjacent pixel values that occur in an image and keeps recording it over the entire image. GLCM demonstrates the occurrence of each gray level at a pixel located at a fixed geometric position relative to each other pixel [16]. Fig. 5 explains how a GLCM is constructed.

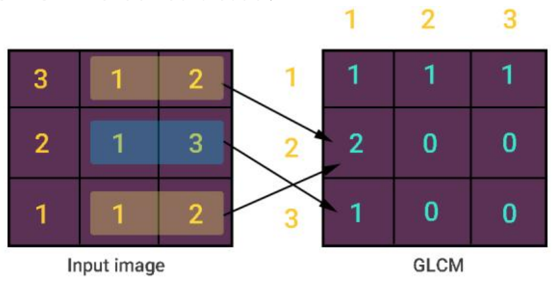


Fig. 5 Gray Level Cooccurrence Matrix (GLCM) construction

Gray-level pixel values 1 and 3 occur only once in the image and thus GLCM records it as one. However, Gray-level pixel values 1 and 2 occur twice in the image and hence GLCM records it as two. This assumes that the adjacency calculation is done only from left-to-right, whereas, in practice, there are four types of adjacency and hence four GLCM matrices are constructed for a single image, and they are as follows: Left-to-Right, Top-to-Bottom, Top Left-to-Bottom Right, and Top Right-to-Bottom Left.

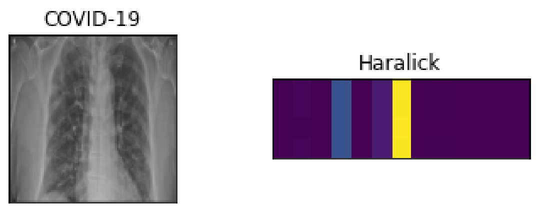


Fig. 6  13 Haralick Textures represented for an X-ray scan image of a COVID-19 positive case.

Thereafter, 14 textural features are computed from the four GLCM matrices, each is based on a statistical theory. The textural features are: Energy (or Square Root of Angular Second Moment), Contrast, Correlation, Sum of Squares: Variance, Inverse Difference Moment, Sum Average, Sum Variance, Sum Entropy, Entropy, Difference Variance, Difference Entropy, Info Measure of Correlation 1, Info Measure of Correlation 2, and Maximal Correlation Coefficient. The equations used to calculate each feature is shown in Fig. 7 but will not be discussed in-depth as it is beyond the scope of this paper.

| | |
|---|---|
| Angular Second Moment | $\sum_i \sum_j p(i,j)^2$ |
| Contrast | $\sum_{n=0}^{N_g-1} n^2 \{\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j)\}, \lvert i-j \rvert = n$ |
| Correlation | $\frac{\sum_i \sum_j (ij)p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y}$ where $\mu_x$ , $\mu_y$ , $\sigma_x$ , and $\sigma_y$ are the means and std. deviations of $p_x$ and $p_y$ , the partial probability density functions |
| Sum of Squares: Variance | $\sum_i \sum_j (i-\mu)^2 p(i,j)$ |
| Inverse Difference Moment | $\sum_i \sum_j \frac{1}{1+(i-j)^2} p(i,j)$ |
| Sum Average | $\sum_{i=2}^{2N_g} i p_{x+y}(i)$ where $x$ and $y$ are the coordinates (row and column) of an entry in the co-occurrence matrix, and $p_{x+y}(i)$ is the probability of co-occurrence matrix coordinates summing to $x+y$ |
| Sum Variance | $\sum_{i=2}^{2N_g} (i - f_8)^2 p_{x+y}(i)$ |
| Sum Entropy | $-\sum_{i=2}^{2N_g} p_{x+y}(i) \log\{p_{x+y}(i)\} = f_8$ |
| Entropy | $-\sum_i \sum_j p(i,j) log(p(i,j))$ |
| Difference Variance | $\sum_{i=0}^{N_g-1} i^2 p_{x-y}(i)$ |
| Difference Entropy | $-\sum_{i=0}^{N_g-1} p_{x-y}(i) \log\{p_{x-y}(i)\}$ |
| Info. Measure of Correlation 1 | $\frac{HXY-HXY1}{\max\{HX,HY\}}$ |
| Info. Measure of Correlation 2 | $(1-\exp[-2(HXY2-HXY)])^{\frac{1}{2}}$ where $HXY = -\sum_i \sum_j p(i,j) \log(p(i,j))$ , $HX$ , $HY$ are the entropies of $p_x$ and $p_y$ , $HXY1 = -\sum_i \sum_j p(i,j) \log\{p_x(i)p_y(j)\} HXY2 = -\sum_i \sum_j p_x(i)p_y(j) \log\{p_x(i)p_y(j)\}$ |
| Max. Correlation Coeff. | Square root of the second largest eigenvalue of $\mathbf{Q}$ where $\mathbf{Q}(i,j) = \sum_k \frac{p(i,k)p(j,k)}{p_x(i)p_y(k)}$ |

Fig. 7  Equations to calculate the 14 Haralick Texture features.

Normally, the feature vector is composed of 13 dimensions since computing the 14th dimension, the Maximal Correlation Coefficient is usually not calculated due to computational instability.

### 4) *Threshold Adjacency Statistics*

Threshold adjacency statistics are generated by first creating a binary image by applying a threshold to the image. The average intensity, μ, of those pixels with intensity at least n is calculated for the image, the cut off n is chosen as intensities below this value are in the general background. The image is then binary threshold-ed to the specified range [μ-n, μ+n]. The range is selected to maximize the visual difference of threshold images.

Mahotas, a computer vision and image processing library for Python, adapts Parameter-free Threshold Adjacency Statistics by Coelho et al. [17], which is an adapted version of the Threshold Adjacency Statistic features (TAS) by Hamilton et al. [18]. Hamilton et al.'s version depends on a manually controlled-two-step binarization of the image. Whereas, for the first step, Coelho et al. use the Ridler–Calvard algorithm to identify a threshold instead of a fixed threshold in Hamilton et al.'s version. The second binarization step involves finding those pixels that fall into a given interval such as [μ−M, μ+M], where μ is the average pixel value of the above-threshold pixel and M is a margin set to the standard deviation of the above threshold pixels. The result of thresholding can be visualized as in Fig. 8. The number of Threshold Adjacency Statistics features extracted for each image by Mahotas is 54.

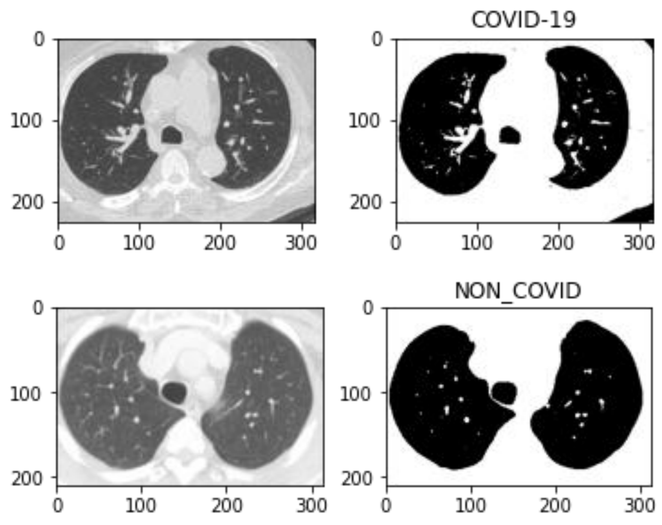


Fig. 8  A CT scan image plotted before (left) and after (right) performing thresholding.

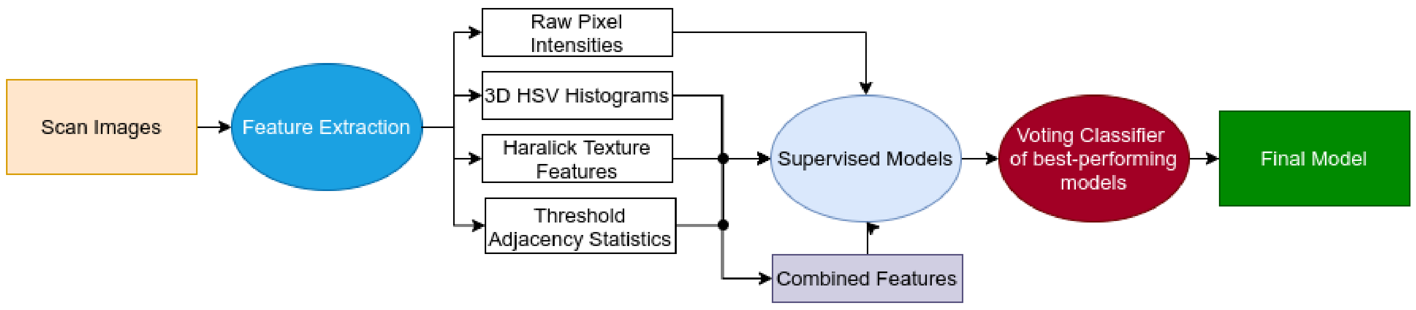


Fig. 9 The process of the proposed models

To summarize, the pseudocode for the algorithm used for the model is as follows: (1) Extract features from images into arrays: raw pixels, color histograms, Haralick features, and threshold adjacency statistics. (2) Merge color histograms, Haralick features, and threshold adjacency statistics into 1 array. (3) Split each feature array from 1 into training data and testing data. (4) Split the array from 2 into training data and testing data. (5) Fit, evaluate, and tune the k-NN, LinearSVC, SVM with RBF kernel, and DT models using the train datasets and test datasets from 3. (6) Fit a voting classifier model with hard voting using the 3 best performing models from 5. (7) Fit, evaluate, and tune the k-NN, LinearSVC, SVM with RBF kernel, and DT models using the training dataset and test dataset from 4. (8) Fit a voting classifier model with hard voting using the 3 best performing models from 7. (9) Compare the performance of each model as well as the time it takes to train and test each model. A diagram of this process is provided in Fig. 9.

## C. *Algorithms*

### 1) *k-Nearest Neighbor Classifier*

KNN is a simple algorithm based on Supervised Learning technique that working by storing different cases and classify other new cases based on distance functions. It has been used in statistical estimation and pattern recognition. It is called a lazy learner algorithm because it depends on storing the data set and perform action on it at the classification time, it does not learn from it immediately. So, basically the algorithm at the training phase just stores the dataset and whenever it gets new data, it classifies it into a category that is much similar to the new data. If K = 1, then the case is simply assigned to the class of its nearest neighbor.

### 2) *Support Vector Machine Classifier*

SVM is a supervised machine learning algorithm that can do both classification and regression, but usually it is being used at classification. SVM algorithm based on plotting data as points that each point represents a specific feature in n-dimensional space where n is the number of features. Then the classification occurs by finding the hyper-plane that differentiates the classes well.

Linear Kernel is used to separate a Linearly separable data. It is used when there are many features in the data set, and it is one of the most common kernels to be used.

Radial Basis Kernel is a kernel function that is used to separate a non-Linearly separable data. The kernel calculates in any d-dimensional space where $d > 1$, so that we can get a quadratic, cubic or any polynomial equation of large degree for our classification/regression line. Since Radial basis using this kernel, it makes the classification line infinitely powerful as it uses exponent which its expansion gives a polynomial equation of infinite power.

### 3) *Random Forest Classifier*

The ensemble method is one of the most popular concepts in the field of machine learning. It is a technique based on dividing one model into more than one sub-model or several repeating models with different experiments or different data in order to reach the best possible accuracy.

The random forest algorithm is an ensemble method where the result of the model is made up of the best result reached by many experiments of different samples within many decision trees models. The algorithm is very effective in classification and regression, as it can create hundreds of decision trees and achieve the best possible output.

The best prediction can be obtained by voting with the best mean, which makes this much better than using only one decision tree, as this algorithm overlooks and greatly reduces the over-fitting problem by accessing the Averaging of all expected results and the closest thing that can be reached.

There are many advantages of random trees. The individual decision trees appear to overfit the training data, but by combining the prediction outcomes from multiple trees, random forests will minimize the problem. This provides a better statistical precision of random forests than a single decision tree. In your dataset, the random forest algorithm will also help you to find features that are significant. It lies at the basis of the Boruta algorithm, which in a dataset selects essential attributes.

Random forest has been used in a variety of applications, such as supplying e-commerce consumers with suggestions for various goods. A random forest algorithm can be used in medicine to recognize the patient's condition by examining the medical record of the patient. It can also be used in the banking industry to accurately identify whether clients are fake or genuine [19].

The first step is to begin selecting random samples from the dataset. The second step is to construct a decision tree for each of the samples that have been prepared from the dataset and assign some predictions to each tree that was made. The third step is to vote for the best possible decision tree. The fourth step is to select the most voted prediction in order to be the final result.

#### *4) Voting Classifier*

A Voting Classifier is a model of machine learning that trains on an ensemble of multiple models and predicts an outcome (class) based on the highest possibility of the class assigned as the output.

It essentially aggregates the results of each classifier passed into the Voting Classifier and based on most votes, predicts the output class. Instead of making individual dedicated models and finding the precision for each of them, the idea is to make a single model that trains these models and forecasts output for each output class based on their cumulative majority of votes.

Two ways of voting are supported by the Voting Classifier: Hard Voting: The predicted result class is a class with the largest majority of votes in hard voting, i.e. the class where each of the classifiers has the highest probability of being predicted. Soft Voting: Based on the average probabilities given to that class, the result class is the predictor for soft voting.

#### *5) MLP Classifier*

Multi-layer Perceptron (MLP) is an algorithm for supervised learning, which learns a function: Where M is the number of dimensions for input and so is the number of dimensions for output, by practicing on a dataset. Provided a set of characteristics and target, A non-linear function approximator for either classification or regression can be taught. It is distinct from logistic regression, in that there can be one or two non-linear layers, called hidden layers, between the input and the output layer. A single hidden layer of MLP with scalar output [20].

A collection of neurons consisting of the leftmost layer, defined as the input nodes, Representing feedback characteristics. With a weighted linear summation, each neuron in the hidden layer converts values from the previous layer. A non-linear g(.) activation function is accompanied by: R—>R - like the hyperbolic tan function. The output layer collects and converts the values from the last hidden layer to output values.

The strengths of the Perceptron multi-layer are the Capability to learn structures that are non-linear and the Real-time opportunity to learn templates. Multi-layer Perceptron (MLP) disadvantages include: MLP has a non-convex lower bound of hidden layers where more than one local minimum occurs. Several hyperparameters, such as the number of hidden nodes, layers, and intervals, need to be calibrated to MLP.

Using Stochastic Gradient Descent, Adam, or L-BFGS MLP trains. Using the gradient of the loss function, Stochastic Gradient Descent (SGD) updates parameters with respect to a parameter that requires adaptation, i.e. Where the learning rate that regulates the step-size in the search for parameter space is. The Loss function used by the network is the one used. L-BFGS is a solver which approximates the Hessian matrix representing a function's second-order partial derivative. In the sense that it is a stochastic optimizer, Adam is like SGD, but can dynamically change the quantity to update parameters based on adaptive predictions of lower-order moments. It also approximates the inverse of the Hessian matrix in order to change parameters.

### D. *Experiment Platform*

The hardware and software environments used in this study are as follows:

Software environment: elementary OS 5.1.7 Hera operating system, Linux 5.4.0-56-generic kernel, Jupyter Notebook 6.1.4 (Python 3.8.3).

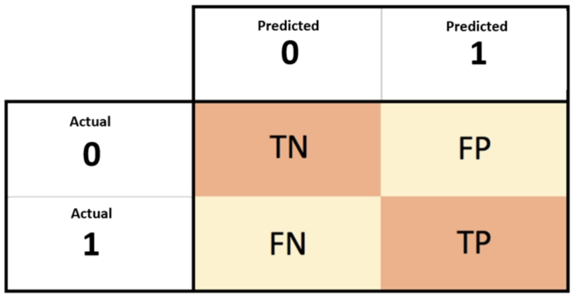
Hardware environment: Quad-Core Intel® Core™ i5-8250U CPU @ 1.60GHz, 6.0 GB RAM.

## III. Results and Discussion

### A. *Performance Evaluation Metrics*

It is critical to evaluate the performance of the classification algorithm after the algorithm performs the classification process. The performance of the classification model is evaluated after it is built to find out how good the model is in predicting the outcome of test data, that is new observations that have been not used to train the model. The model prediction accuracy and prediction errors are calculated using the test dataset. Since the outcomes of the observations in the test dataset are known beforehand, the performance of the predictive model can easily be determined by comparing the predicted outcomes against the known outcomes. The confusion matrix is used to check the performance of the classification algorithm proposed in this paper. The confusion matrix offers crucial information regarding the performance of the model.

A confusion matrix calculates how many observations were correctly or incorrectly classified and shows the number of correct and incorrect predictions categorized by type of outcome [21]. Correct predictions are indicated by the diagonal elements of the confusion matrix, while the off diagonals represent incorrect predictions, as shown in Fig. 10. The following is the explanation of each call from the confusion matrix:



|  | Predicted 0 | Predicted 1 |
|---|---|---|
| Actual 0 | TN | FP |
| Actual 1 | FN | TP |

Fig. 10 A confusion matrix as represented in sklearn.

True positives (TP): the cases in which the model predicted that the chest scan image is for an individual who is COVID-19 positive and they were, in reality, COVID-19 positive.

True negatives (TN): the cases in which the model predicted that the chest scan image is for an individual who is COVID-19 negative and they were, in reality, COVID-19 negative.

False positives (FP): the cases in which the model predicted that the chest scan image is for an individual who is COVID-19 positive, but the individual didn’t actually have COVID-19 (also known as a Type I error).

False negatives (FN): the cases in which the model predicted that the chest scan image is for an individual who is COVID-19 negative, but the individual actually had COVID-19 (also known as a Type II error).

Different characteristics of the classification performance metrics can be calculated using information from the confusion matrix. The performance evaluation metrics commonly used in literature are accuracy, specifity, sensitivity, and precision. These metrics are computed using Eq. 1–4, respectively [22]. In this study, the training datasets and test datasets are randomly assigned to test the model comprehensively.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{2}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{3}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Precision is the proportion of true COVID-19 positive cases among all the cases that have been predicted to be COVID-19 positive by the model.

Sensitivity (or Recall), is the True Positive Rate (TPR) or the proportion of identified COVID-19 positive cases among all the true COVID-19 positive population. Specificity measures the True Negative Rate (TNR), that is the proportion of identified COVID-19 negative cases among all the COVID-19 negative population. The overall classification accuracy is the rate of the proportion of observations that have been correctly classified, as in the ratio of correctly identified positive COVID-19 cases and correctly identified negative COVID-19 cases, to the entire population.

## B. *Experimental Results and Discussion*

Each model is trained separately by randomly splitting the corresponding datasets into training datasets, used to train the model, and testing datasets, used for testing the model's performance. For the CT scan images dataset, a total of 2,471 images, with a training subset of 1,976 images and a testing subset of 495 images (80% and 20%) is used. As for the X-ray scan images dataset, a total of 1,227 images, with a training subset of 981 images and a testing subset of 246 images (80% and 20%) is used.

After extracting each of the following features: raw pixel intensities, HSV color histograms, Haralick features, threshold adjacency statistics, each feature is used separately to train the supervised learning models used in this paper: k-NN, Linear SVM, SVM with RBF kernel, Random Forest, and Voting Classifier with Hard voting using the 'most-accurate' three models. Afterward, HSV Color Histograms, Haralick Texture features, and Threshold Adjacency Statistics features are all combined (CH, HT, & TAS) and used to train each of the previously mentioned models. Due to its poor performance on X-ray scan images, the raw pixel intensities are not used for the X-ray model. GridSearchCV is used to find the best parameters for the SVM model using the RBF kernel. Moreover, to determine the best k to use for the k-NN models, this study uses a graph plotting each k-NN model's performance against different k values ranging from 1 to 9, and to avoid overfitting the model, even values, as well as 1, are discarded, so only odd values (3 and above) are used for 'k' since the model is used to classify data into one of two classes (also known as Binary Classification).

The following section shows the training accuracies and testing accuracies of each of the models trained for each of the features extracted as well as the combined features, using the CT scan images dataset and X-ray scan images dataset.

### 1) *Using CT dataset*

Dataset: 2,471 images, 241.2 MB, 80% used as a training dataset.

TABLE I
SUMMARY OF THE CT MODEL

| Feature | # of dimensions | Size | Highest Accuracy |
|---|---|---|---|
| Raw Pixel Intensities | 3600 | 8.69MB | 95.6% |
| HSV Color Histograms | 692 | 6.68MB | 97.0% |
| Haralick Textures | 13 | 0.25MB | 94.7% |
| TAS Features | 54 | 1.04MB | 93.5% |
| Combined Features (CH, HT, & TAS) | 759 | 14.65MB | 98.2% |

Table. I shows that the Random Forest Classifier is the best performing algorithm on the training dataset, however, the structured Voting Classifier with Hard voting using the best-performing classifiers: k-NN, RBF SVM, and Random Forest, was the most accurate on the testing dataset with a mean 95.56% for all the extracted features. The maximum accuracy for the Voting Classifier on the testing dataset is 98.20% and is obtained using the combined features to train each of the three aforementioned classifiers used for the Voting Classifier. The same structured Voting Classifier has an accuracy of 99.78% on the training dataset. The following section summarizes the different performance metrics used to evaluate the structured algorithm using the Voting Classifier for the CT scan images model. Appendix A includes charts of the training and testing accuracies of the different classifying algorithms on the CT dataset.

TABLE II
VOTING CLASSIFIER COMPOSITION IN THE CT MODEL

| **Voting Classifier on Combined Features (CH, HT, & TAS)** | |
|---|---|
| *Algorithm* | *Parameters* |
| k-NN | n_neighbors=5 |
| SVM | kernel = 'RBF', C = 100, gamma = 'scale' |
| Random Forest | default parameters |

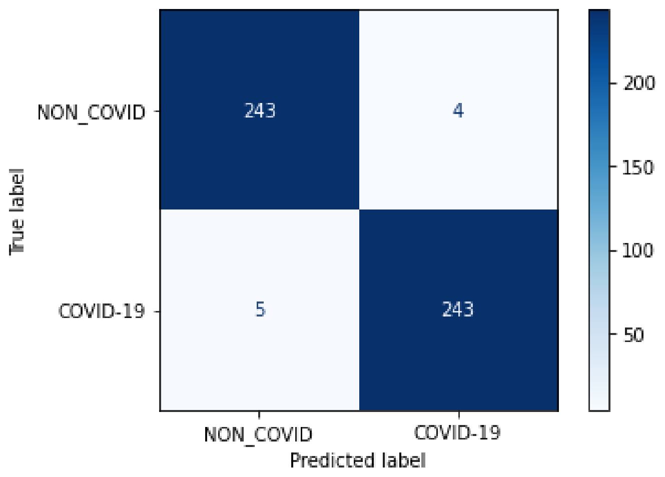


Fig. 11 Confusion matrix of the Voting Classifier on the combined features of the CT model.

TABLE III
CLASSIFICATION REPORT OF CT MODEL'S VOTING CLASSIFIER

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| COVID-19 | 0.99 | 0.98 | 0.98 | 248 |
| NON_COVID | 0.98 | 0.99 | 0.98 | 247 |
| accuracy | | | 0.98 | 495 |

Voting Classifier Scores on the CT model's Combined Features

| Train Accuracy | Test Accuracy | Specificity | Sensitivity | Precision |
|---|---|---|---|---|
| 100% | 98.18% | 98.38% | 97.98% | 98.38% |

2) *Using X-ray dataset*

Dataset: 1,227 images, 287.0 MB, 80% used as a training dataset.

TABLE IV
SUMMARY OF THE X-RAY MODEL

| Feature | # of dimensions | Size | Highest Accuracy |
|---|---|---|---|
| HSV Color Histograms | 692 | 3.32MB | 95.1% |
| Haralick Textures | 13 | 0.12MB | 97.6% |
| TAS Features | 54 | 0.52MB | 98.8% |
| Combined Features (CH, HT, & TAS) | 759 | 7.28MB | 99.6% |

Table IV shows that the Random Forest Classifier is the best performing algorithm on the training dataset, however, the structured Voting Classifier with Hard voting using the best-performing classifiers: Linear SVM, RBF SVM, and Random Forest, was the most accurate on the testing dataset with a mean 97.45% for all the extracted features. The maximum accuracy for the Voting Classifier on the testing dataset is 99.6% and is obtained using the combined features to train each of the three aforementioned classifiers used for the Voting Classifier. The same structured Voting Classifier has an accuracy of 100% on the training dataset. The following section summarizes the different performance metrics used to evaluate the structured algorithm using the Voting Classifier for the X-ray scan images model. Appendix B includes charts of the training and testing accuracies of the different classifying algorithms on the CT dataset.

TABLE V
VOTING CLASSIFIER COMPOSITION IN THE X-RAY MODEL

| **Voting Classifier on Combined Features (CH, HT, & TAS)** | |
|---|---|
| *Algorithm* | *Parameters* |
| SVM | kernel = 'linear' |
| SVM | kernel = 'RBF', C = 10, gamma = 'scale' |
| Random Forest | default parameters |

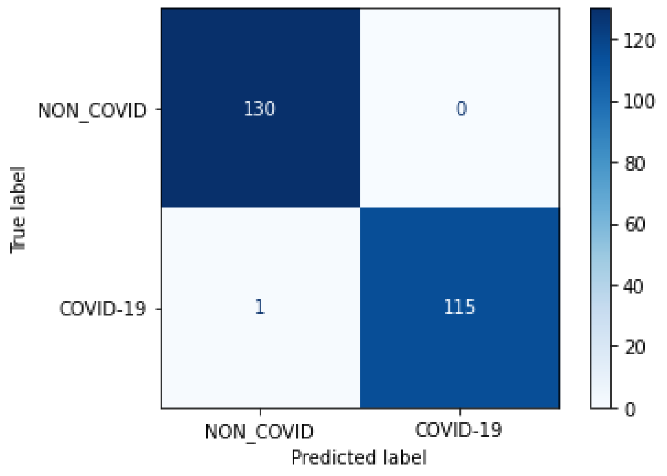


Fig. 12 Confusion matrix of the Voting Classifier on the combined features of the X-ray model.

TABLE VI
CLASSIFICATION REPORT OF X-RAY MODEL'S VOTING CLASSIFIER

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| COVID-19 | 1.00 | 0.99 | 1.00 | 116 |
| NON_COVID | 0.99 | 1.00 | 1.00 | 130 |
| accuracy | | | 1.00 | 246 |

Voting Classifier Scores on the X-ray model's Combined Features

| Train Accuracy | Test Accuracy | Specificity | Sensitivity | Precision |
|---|---|---|---|---|
| 100% | 99.59% | 99.24% | 100% | 99.14% |

### C. Time Consumption

After defining the best adjustable parameters in each algorithm, each classification algorithm developed is tested of the CPU time metric for predictor model training and the CPU time for testing the classification model with testing data instances. Appendix C shows the test results of CPU time for training and testing each algorithm on the CT and X-ray scan images datasets, respectively.

With the results presented in Appendix C, extracting the Haralick Textures and Threshold Adjacency Statistics features from CT and X-ray scan images datasets and using them to train and test the different classifiers yielded the shortest training and testing times across all different classifiers. However, using Raw Pixel Intensities features extracted from CT and X-ray scan images datasets produced the longest execution time for all the classifiers. This is expected since Haralick Textures and Threshold Adjacency Statistics features have the lowest dimensionality and size while Raw Pixel Intensities have the highest dimensionality and largest size of training and testing data.

TABLE VII
TOTAL EXECUTION TIME

**CT Dataset**

| Feature | k-NN | L. SVM | RBF SVM | RF | VC |
|---|---|---|---|---|---|
| **RPI** | 15.061 | 24.024 | 40.770 | 9.749 | 64.440 |
| **CH** | 2.748 | 3.283 | 1.958 | 1.861 | 6.499 |
| **HT** | 0.046 | 0.122 | 0.166 | 0.780 | 0.993 |
| **TAS** | 0.151 | 0.346 | 0.373 | 1.442 | 1.941 |
| **CH, HT, & TAS** | 3.049 | 2.790 | 2.145 | 2.169 | 7.844 |

**X-ray Dataset**

| Feature | k-NN | L. SVM | RBF SVM | RF | VC |
|---|---|---|---|---|---|
| **CH** | 0.680 | 1.079 | 0.775 | 1.446 | 2.881 |
| **HT** | 0.023 | 0.039 | 0.044 | 0.529 | 0.549 |
| **TAS** | 0.055 | 0.029 | 0.038 | 0.666 | 0.765 |
| **CH, HT, & TAS** | 0.726 | 0.285 | 0.732 | 0.987 | 1.963 |

Furthermore, the classification algorithm that produced the shortest training time for both datasets is k-NN, but Random Forest Classifier yields the shortest testing time on both datasets. Table VII shows the total execution time, that is the sum of training time and testing time, for each of the classification algorithms on both datasets. The algorithm with the shortest total execution time on the CT dataset is Random Forest, whereas SVM with Linear kernel produced the shortest total execution time on the X-ray dataset. Predictably, the longest total execution time is produced when using the Voting Classifier for either dataset, since it depends on first fitting the three different classifiers of which it is structured. However, the increase in execution time for the Voting Classifier is not excessively large as it only took the structured classifier a total execution time of 7.844s in the case of the CT model, and 1.963s for the X-ray model. To calculate the total elapsed time for the model, the feature extraction time is added to the total execution time for the structured Voting Classifier. Extracting all the features from the datasets takes an average of 480s for CT scan images and 499s for X-ray scan images. Therefore, the total execution time is estimated to be within 9 minutes for each of the respective datasets using the hardware and software environments specified in the 'Experiment Platform' section.

### D. Web Application

This paper uses Anvil to create a web application for the classification model. Anvil is a platform for building and hosting full-stack web apps written entirely in Python. The classification model is connected to the Anvil app using the Anvil Uplink, which creates a bidirectional connection with the Python code to make it behave like a Server Module, so that functions in it can be called from the app using anvil.server.call, and calling anvil.server.call is possible into the app from the Uplink. In this study, the Uplink is connected to the Jupyter Notebook to create an Anvil front-end for the classification model.

The web application asks the user to upload an image of either a chest CT scan image or an X-ray scan image, depending on which model is trained and running, then passes this image to several classifiers, each predicts the class of this scan image, either COVID-19 or NON_COVID. The web application returns the following outputs: k-NN Probability of COVID-19, RBF SVM Probability of COVID-19, LinearSVC Probability of COVID-19, Random Forest Probability of COVID-19, MLP

Probability of COVID-19, and the Voting Classifier with Hard Voting Result (depending on the predictions of the three models of which it is composed). The web app is tested using an unbiased subset from each of the corresponding CT and X-ray datasets that the model was neither trained on nor initially tested for. The subset offers unbiased novel inputs. Appendices D & E show the results of testing the web app on CT and X-ray models, respectively, using the novel inputs. Appendix F shows the growing confidence of the X-ray model with the progress of COVID-19 pneumonia in the lungs of a patient.

## IV. Conclusions

In this paper, two novel models are proposed for the detection of COVID-19 in CT and X-ray scan images using supervised machine learning classifying algorithms. The first model is used for CT scan images and is able to determine whether a given chest CT image of a patient is infected with COVID-19 or not with 98.2% average accuracy. The second model is used for X-ray scan images and is able to detect COVID-19 in a patient's chest X-ray scan image with 99.6% average accuracy. The proposed models are based on image feature extraction methods to extract raw pixel intensities, HSV color histograms, Haralick texture features, and threshold adjacency statistics. Therefore, the models require relatively low computational power for training and testing compared to the currently available methods in the literature. A total of 2,471 CT and 1,227 X-ray scan images are used to train and test the models. Experimental results show the effectiveness of the proposed models. This paper aims to offer an efficient low-cost alternative to PCR tests to detect COVID-19. It is believed that the models proposed in this paper, due to their efficiency and flexibility, can be readily used in practice to help physicians in diagnosing the COVID-19 disease, particularly in rural areas where patients cannot afford the PCR test or in countries where the necessary resources to conduct widespread testing are not available.

## V. Acknowledgment

The authors acknowledge the contributions of Eduardo Soares, Plamen Angelov, Sarah Biaso, Michele Higa Froes, and Daniel Kanda Abe, Muhammad E. H. Chowdhury, Tawsifur Rahman, Amith Khandakar, Rashid Mazhar, Muhammad Abdul Kadir, Zaid Bin Mahbub, Khandakar Reajul Islam, Muhammad Salman Khan, Artificial Intelligence in Healthcare, Atif Iqbal, Nasser Al Emadi, Mamun Bin Ibne Reaz, and Mohammad Tariqul Islam for creating the datasets used in this study and offering their datasets for public access.

## VI. References

[1] CDC, "Coronavirus Disease 2019 (COVID-19)," *Centers for Disease Control and Prevention*, Feb. 11, 2020. https://www.cdc.gov/coronavirus/2019-ncov/faq.html (accessed Jan. 04, 2021).

[2] "Archived: WHO Timeline - COVID-19." https://www.who.int/news/item/27-04-2020-who-timeline---covid-19 (accessed Jan. 04, 2021).

[3] "COVID-19 Map - Johns Hopkins Coronavirus Resource Center." https://coronavirus.jhu.edu/map.html (accessed Jan. 04, 2021).

[4] W. Guan *et al.*, "COVID-19: Antiviral Agents, Antibody Development and Traditional Chinese Medicine," *Virol. Sin.*, Sep. 2020, doi: 10.1007/s12250-020-00297-0.

[5] A. Mullard, "How COVID vaccines are being divvied up around the world," *Nature*, Nov. 2020, doi: 10.1038/d41586-020-03370-6.

[6] "Rich countries hoarding Covid vaccines, says People's Vaccine Alliance," *BBC News*, Dec. 09, 2020.

[7] S. B. H. editor, "Nine out of 10 in poor nations to miss out on inoculation as west buys up Covid vaccines," *The Guardian*, Dec. 09, 2020.

[8] "Medical Image Processing in Detection of Abdomen Diseases | SpringerLink." https://link.springer.com/chapter/10.1007%2F978-981-15-1100-4_6 (accessed Jan. 13, 2021).

[9] E. Soares, P. Angelov, S. Biaso, M. H. Froes, and D. K. Abe, "SARS-CoV-2 CT-scan dataset: A large dataset of real patients CT scans for SARS-CoV-2 identification," *medRxiv*, p. 2020.04.24.20078584, May 2020, doi: 10.1101/2020.04.24.20078584.

[10] M. E. H. Chowdhury *et al.*, “Can AI Help in Screening Viral and COVID-19 Pneumonia?,” *IEEE Access*, vol. 8, pp. 132665–132676, 2020, doi: 10.1109/ACCESS.2020.3010287.

[11] M. S. Nixon and A. S. Aguado, *Feature extraction and image processing*, 1st ed. Oxford ; Boston: Newnes, 2002.

[12] A. R. Smith, “Color gamut transform pairs,” in *Proceedings of the 5th annual conference on Computer graphics and interactive techniques - SIGGRAPH ’78*, Not Known, 1978, pp. 12–19, doi: 10.1145/800248.807361.

[13] M. K. Agoston, *Computer graphics and geometric modeling: implementation and algorithms*. London: Springer, 2005.

[14] “Changing Colorspaces — OpenCV-Python Tutorials 1 documentation.” https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html (accessed Jan. 03, 2021).

[15] R. M. Haralick, K. Shanmugam, and I. Dinstein, “Textural Features for Image Classification,” *IEEE Trans. Syst. Man Cybern.*, vol. SMC-3, no. 6, pp. 610–621, Nov. 1973, doi: 10.1109/TSMC.1973.4309314.

[16] N. Zayed and H. A. Elnemr, “Statistical Analysis of Haralick Texture Features to Discriminate Lung Abnormalities,” *International Journal of Biomedical Imaging*, Oct. 08, 2015. https://www.hindawi.com/journals/ijbi/2015/267807/ (accessed Jan. 11, 2021).

[17] L. P. Coelho *et al.*, “Structured Literature Image Finder: Extracting Information from Text and Images in Biomedical Literature,” in *Linking Literature, Information, and Knowledge for Biology*, Berlin, Heidelberg, 2010, pp. 23–32, doi: 10.1007/978-3-642-13131-8_4.

[18] N. A. Hamilton, R. S. Pantelic, K. Hanson, and R. D. Teasdale, “Fast automated cell phenotype image classification,” *BMC Bioinformatics*, vol. 8, no. 1, p. 110, Mar. 2007, doi: 10.1186/1471-2105-8-110.

[19] A. Cutler, D. R. Cutler, and J. R. Stevens, “Random Forests,” in *Ensemble Machine Learning: Methods and Applications*, C. Zhang and Y. Ma, Eds. Boston, MA: Springer US, 2012, pp. 157–175.

[20] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Subsequent edition. Upper Saddle River, N.J: Prentice Hall, 1998.

[21] R. Bhagwat, M. Abdolahnejad, M. Moocarme, and an O. M. C. Safari, *Applied Deep Learning with Keras*. 2019.

[22] N. Japkowicz and M. Shah, *Evaluating learning algorithms: a classification perspective*. New York: Cambridge University Press, 2014.

## VII. Appendix A

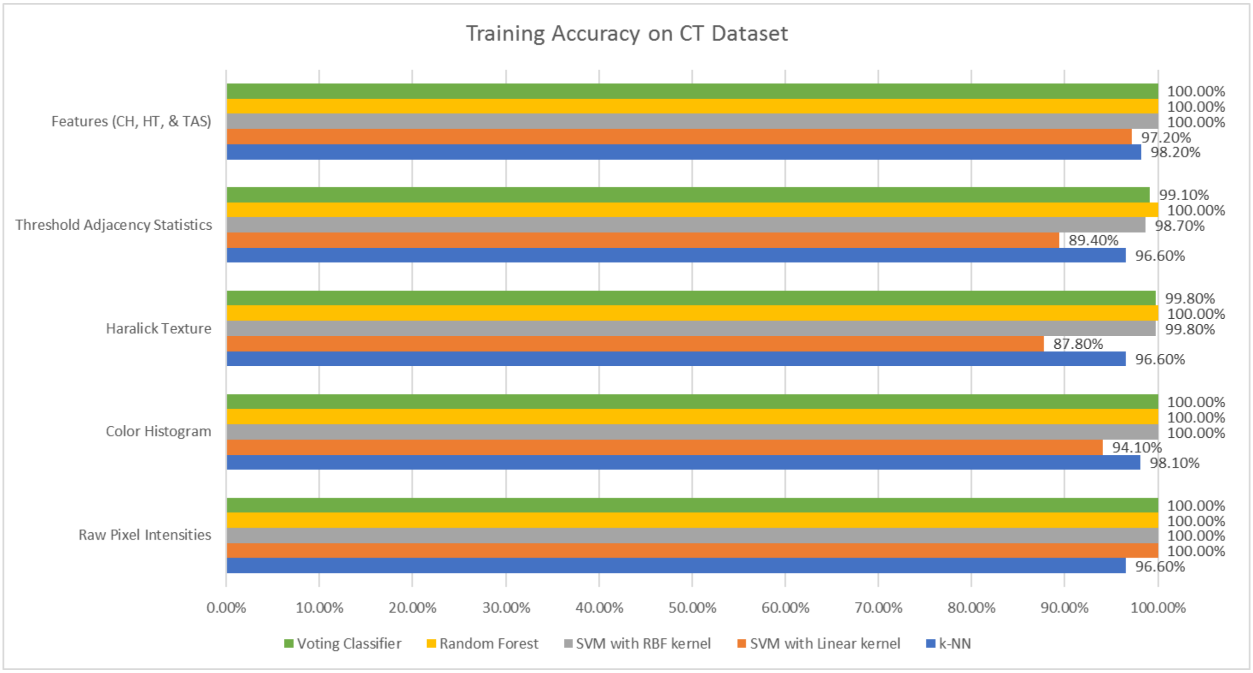


Fig. 13 Training accuracies of the classifying algorithms on the CT dataset

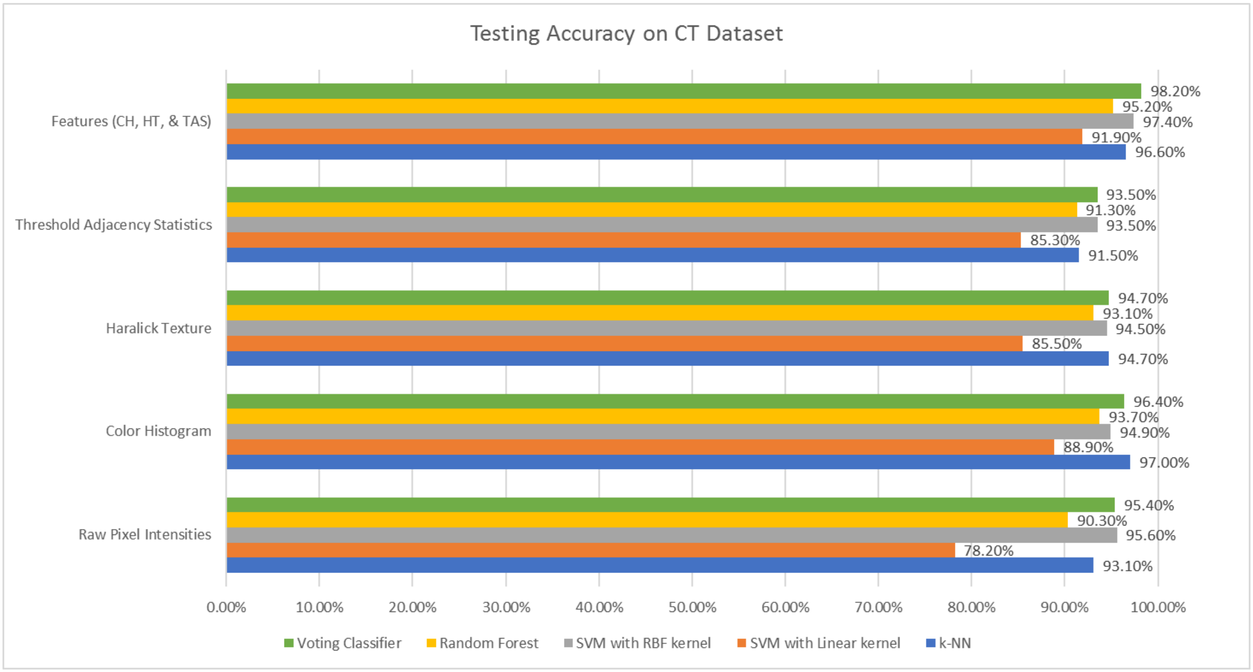


Fig. 14 Testing accuracies of the classifying algorithms on the CT dataset

## VIII. APPENDIX B

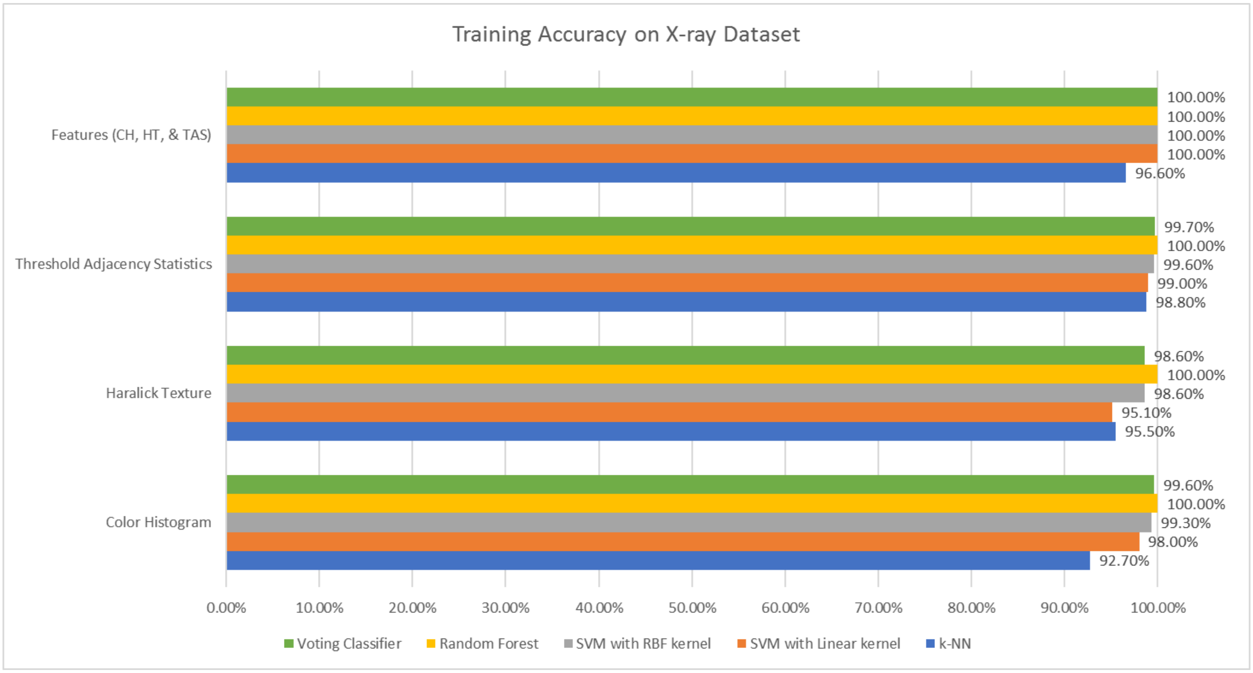


Fig. 15 Training accuracies of the classifying algorithms on the X-ray dataset

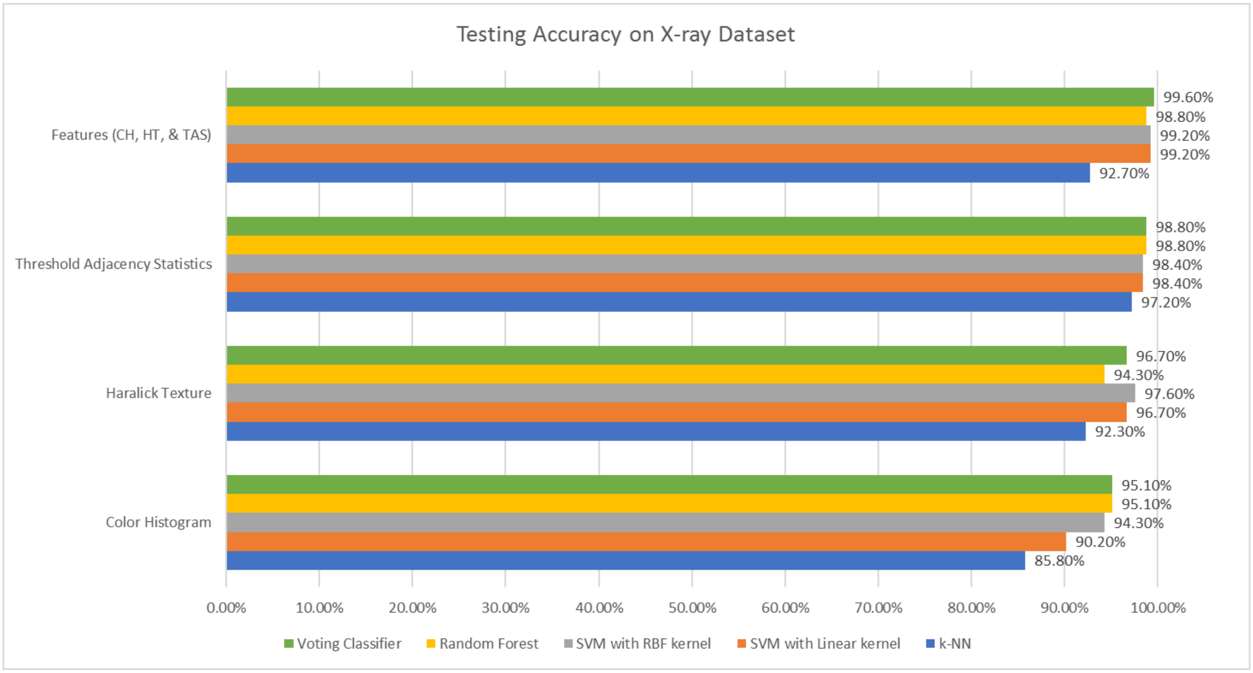


Fig. 16 Testing accuracies of the classifying algorithms on the X-ray dataset

## IX. Appendix C

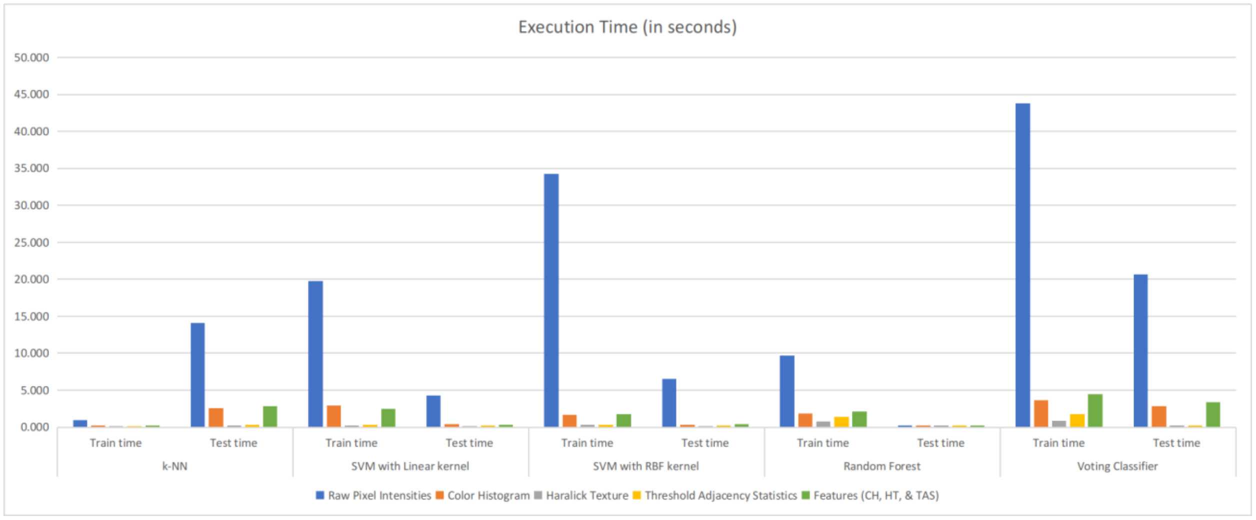


Fig. 17 Execution time of the classifying algorithms on the CT dataset in seconds

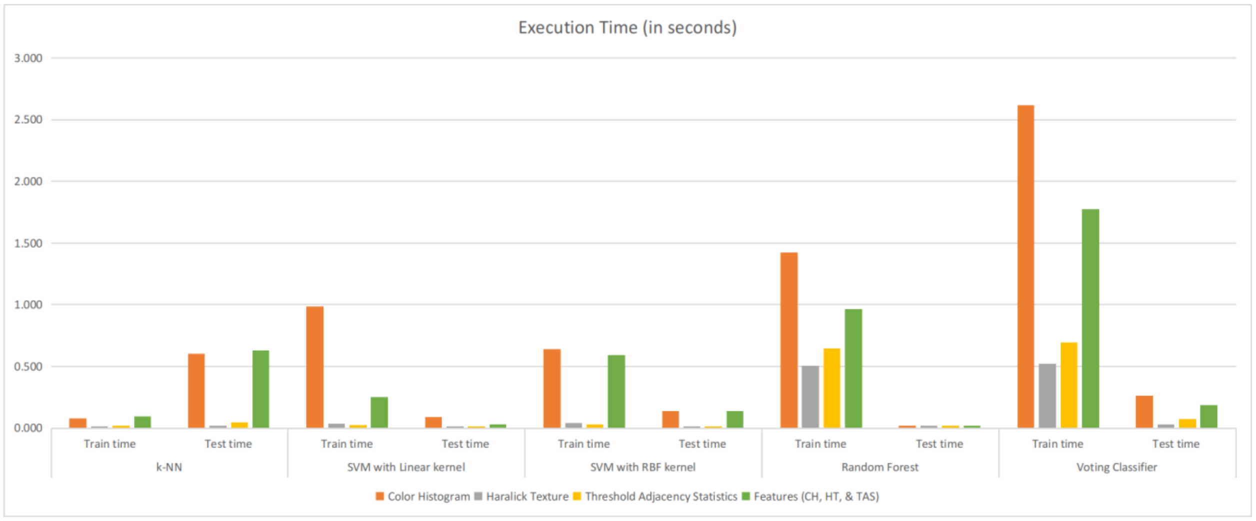
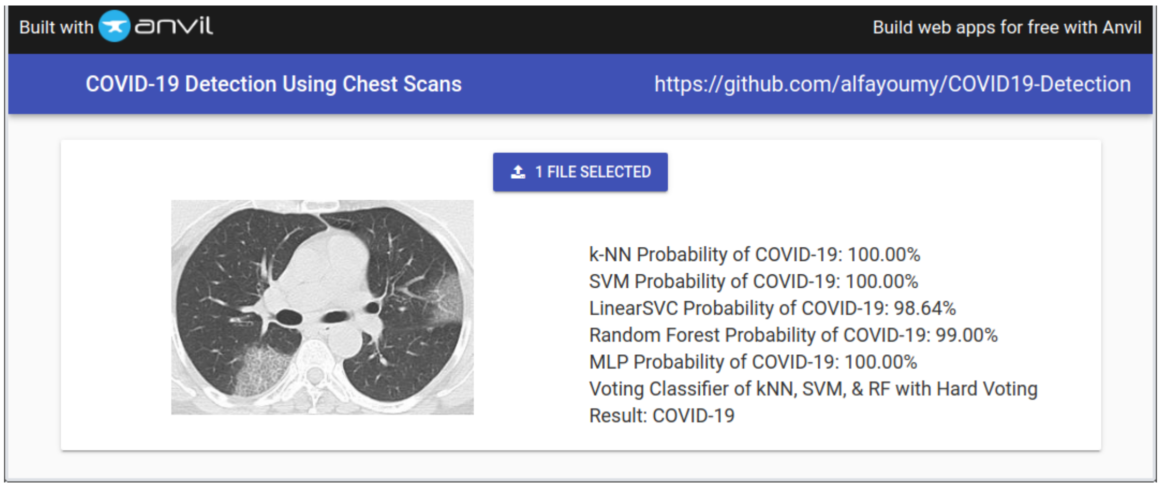


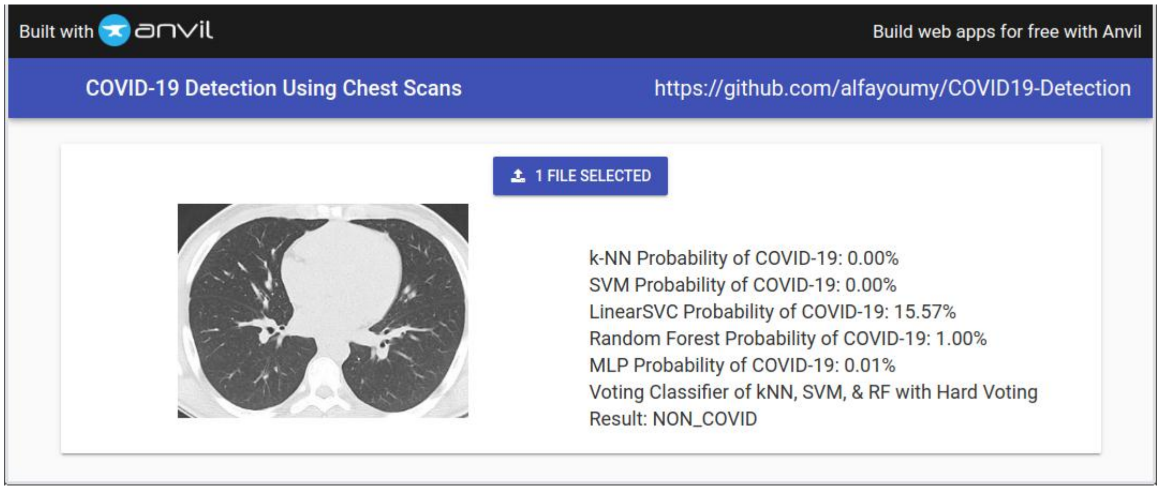
Fig. 18 Execution time of the classifying algorithms on the X-ray dataset in seconds

## X. Appendix D



COVID-19 Detection Using Chest Scans https://github.com/alfayoumy/COVID19-Detection

k-NN Probability of COVID-19: 100.00%
SVM Probability of COVID-19: 100.00%
LinearSVC Probability of COVID-19: 98.64%
Random Forest Probability of COVID-19: 99.00%
MLP Probability of COVID-19: 100.00%
Voting Classifier of kNN, SVM, & RF with Hard Voting
Result: COVID-19

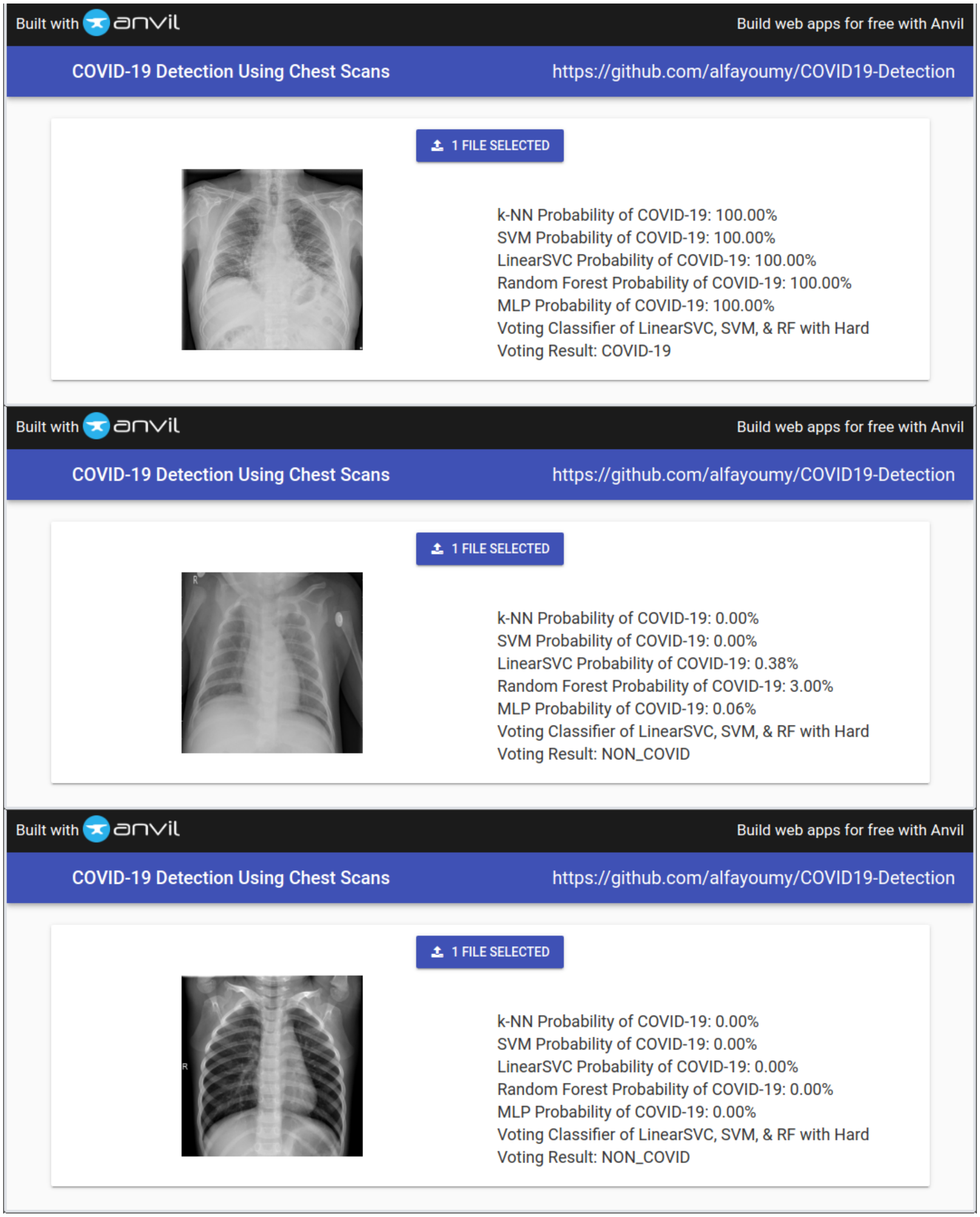


COVID-19 Detection Using Chest Scans https://github.com/alfayoumy/COVID19-Detection

k-NN Probability of COVID-19: 0.00%
SVM Probability of COVID-19: 0.00%
LinearSVC Probability of COVID-19: 15.57%
Random Forest Probability of COVID-19: 1.00%
MLP Probability of COVID-19: 0.01%
Voting Classifier of kNN, SVM, & RF with Hard Voting
Result: NON_COVID

Fig. 19 The proposed model tested through the web app on a CT scan image of a COVID-19 positive case (above) and a CT scan image of a COVID-19 negative case (below)

## XI. Appendix E


Built with anvil
Build web apps for free with Anvil
COVID-19 Detection Using Chest Scans
https://github.com/alfayoumy/COVID19-Detection
1 FILE SELECTED

k-NN Probability of COVID-19: 100.00%
SVM Probability of COVID-19: 100.00%
LinearSVC Probability of COVID-19: 100.00%
Random Forest Probability of COVID-19: 100.00%
MLP Probability of COVID-19: 100.00%
Voting Classifier of LinearSVC, SVM, & RF with Hard
Voting Result: COVID-19

Built with anvil
Build web apps for free with Anvil
COVID-19 Detection Using Chest Scans
https://github.com/alfayoumy/COVID19-Detection
1 FILE SELECTED

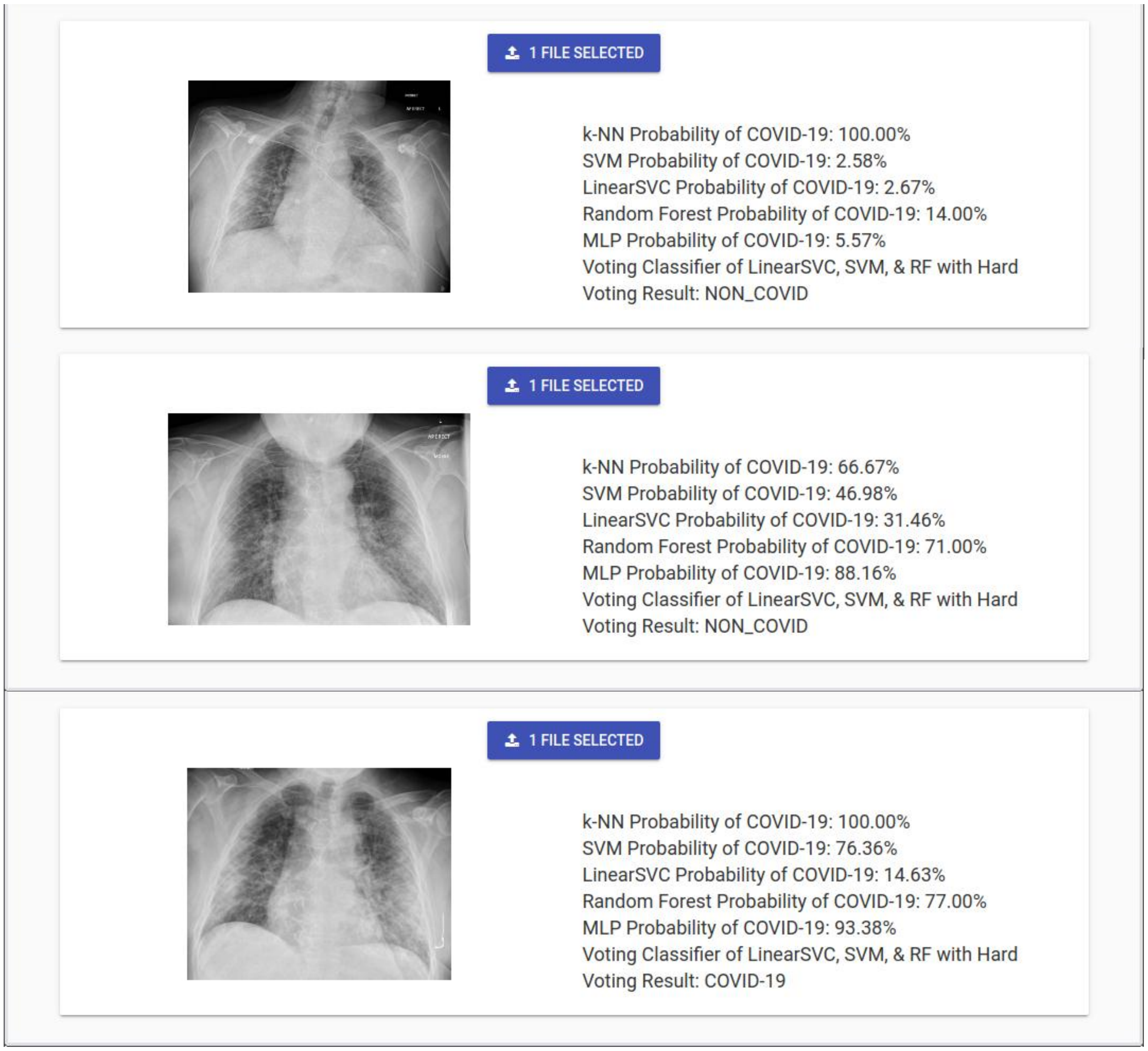
k-NN Probability of COVID-19: 0.00%
SVM Probability of COVID-19: 0.00%
LinearSVC Probability of COVID-19: 0.38%
Random Forest Probability of COVID-19: 3.00%
MLP Probability of COVID-19: 0.06%
Voting Classifier of LinearSVC, SVM, & RF with Hard
Voting Result: NON_COVID

Built with anvil
Build web apps for free with Anvil
COVID-19 Detection Using Chest Scans
https://github.com/alfayoumy/COVID19-Detection
1 FILE SELECTED

k-NN Probability of COVID-19: 0.00%
SVM Probability of COVID-19: 0.00%
LinearSVC Probability of COVID-19: 0.00%
Random Forest Probability of COVID-19: 0.00%
MLP Probability of COVID-19: 0.00%
Voting Classifier of LinearSVC, SVM, & RF with Hard
Voting Result: NON_COVID

Fig. 20 The proposed model tested through the web app on a CT scan image of a COVID-19 positive case (above), a CT scan image of a viral pneumonia case (middle), and a CT scan image of a normal lung (below).

## XII. APPENDIX F



k-NN Probability of COVID-19: 100.00%
SVM Probability of COVID-19: 2.58%
LinearSVC Probability of COVID-19: 2.67%
Random Forest Probability of COVID-19: 14.00%
MLP Probability of COVID-19: 5.57%
Voting Classifier of LinearSVC, SVM, & RF with Hard
Voting Result: NON_COVID

k-NN Probability of COVID-19: 66.67%
SVM Probability of COVID-19: 46.98%
LinearSVC Probability of COVID-19: 31.46%
Random Forest Probability of COVID-19: 71.00%
MLP Probability of COVID-19: 88.16%
Voting Classifier of LinearSVC, SVM, & RF with Hard
Voting Result: NON_COVID

k-NN Probability of COVID-19: 100.00%
SVM Probability of COVID-19: 76.36%
LinearSVC Probability of COVID-19: 14.63%
Random Forest Probability of COVID-19: 77.00%
MLP Probability of COVID-19: 93.38%
Voting Classifier of LinearSVC, SVM, & RF with Hard
Voting Result: COVID-19

Fig. 21 The proposed model tested through the web app on a CT scan image of a COVID-19 positive 70-year-old male patient on the 1st day (above), 3rd day (middle), and 7th day (below). (Scan image source: Chesterfield Royal Hospital)