ACCSCIENCE PUBLISHING

Artificial Intelligence in Health

ORIGINAL RESEARCH ARTICLE

Predicting mortality outcomes in individual COVID-19 patients using machine learning algorithms

Nikolaos Kourmpanis*, Joseph Liaskos, Emmanouil Zoulias, and John Mantas

Laboratory of Health Informatics, Department of Public Health, Faculty of Nursing, National and Kapodistrian University of Athens, Athens, Greece

Abstract

In late 2019, the COVID-19 disease emerged, caused by the SARS-CoV-2 virus, and has since spread worldwide, becoming a global pandemic and resulting in almost seven million deaths to date. In addressing this global crisis, artificial intelligence has played a crucial role, particularly through the development of predictive models using machine learning algorithms, which have been successfully applied to solving a multitude of problems across multiple scientific fields. The purpose of this paper is to identify the model, or models, with the highest accuracy in predicting a COVID-19 patient's mortality outcome by comparing their performance metrics. Different ML methods employed in model development include logistic regression, decision trees, random forest, eXtreme gradient boosting (XGBoost), multi-layer perceptrons, and the k-nearest neighbors. The metrics used for the comparison of these models were accuracy, precision-recall, F1 score, area under the receiver operating characteristic curve (AUC-ROC), and runtime. The data used comprised the clinical characteristics and histories of 12,425,179 individuals who attended health facilities in Mexico. Following a comprehensive evaluation, the XGBoost model achieved the highest overall score across all metrics. It scored 93.76% in precision, 95.47% in recall, 91.13% in F1-score, 97.86% in AUC-ROC, and had a runtime of 6.67306 s. Therefore, XGBoost was determined to be the preferred method for predicting the mortality outcome of COVID-19 patients.

Keywords: COVID-19; Pandemic; Machine learning; Classification algorithm

***Corresponding author:**
Nikolaos Kourmpanis
(nikos.kourbanis@gmail.com)

Citation: Kourmpanis N, Liaskos J, Zoulias E, Mantas J. Predicting mortality outcomes in individual COVID-19 patients using machine learning algorithms. *Artif Intell Health*. 2024;1(3):31-52. doi: 10.36922/aih.2591

Received: December 30, 2023

Accepted: May 9, 2024

Published Online: July 22, 2024

1. Introduction

Coronavirus disease 2019 (COVID-19)[1] was first identified in the Chinese city of Wuhan in December 2019. On January 30, 2020, the World Health Organization (WHO) classified this outbreak as a Public Health Emergency of International Concern, and on March 11, 2020, it declared COVID-19 a pandemic.[2,3] As of December 13, 2023, there have been 772,386,069 confirmed cases of COVID-19 and 6,987,222 deaths reported to the WHO.[4] COVID-19 is caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2),[5] an RNA virus with a spherical shape and a genome composed of a positive-polarity single-stranded RNA.[6] The virus enters human cells through the

angiotensin-converting enzyme 2 (ACE2) receptor. The viral spike protein first attaches to ACE2, and then the membrane enzyme TMPRSS2 cleaves the spike protein, exposing fusion peptides that facilitate fusion with the cell membrane.[7,8] SARS-CoV-2 is the ninth documented coronavirus to infect humans and the seventh identified in the past 20 years.[9,10] Viruses related to SARS-CoV-2 have been documented in bats and pangolins in multiple locations in Southeast Asia, including China, Thailand, Cambodia, and Japan,[11,12] with serological evidence of viral infections in pangolins for more than a decade.[13] Notably, SARS-CoV-2 is primarily transmitted between people through close contact.

The explosion in the number of infections and deaths has led to global efforts to control and curb the disease's spread and associated mortality. One research field that has had a major positive impact on our understanding and fighting the pandemic is machine learning (ML). ML was developed as a tool for data analysis and pattern recognition.[14] ML algorithms process known data and represent it in mathematical ways.[14] During the pandemic, ML studies have assisted in diagnosing and predicting the severity of illness and mortality of COVID-19,[15,16] predicting future mutations of SARS-CoV-2,[17] and promoting the rapid development of therapeutic strategies such as effective vaccines[18] against the virus.

The present study focuses on the prediction of COVID-19 patient mortality using risk factors such as health conditions, habits, and others. Many factors increase the severity of COVID-19 disease, which may even result in the death of the sufferer. A key risk indicator is age, as older people are more likely to get seriously ill from COVID-19. Over 81% of deaths from COVID-19 are among people over the age of 65. The number of deaths among people aged over 65 years is 80 times higher than the number of deaths among people aged 18 – 29 years.[19] There are also medical conditions that increase the severity of the disease, such as heart disease,[19] type I and II diabetes,[20,21] chronic lung diseases,[22] and obesity.[23] In addition, smoking can negatively affect the severity of COVID-19 illness, as it is one of the risk factors for the development and exacerbation of multiple respiratory diseases.[24,25]

The data used in this study were provided by the Epidemiological Surveillance System for Respiratory Diseases under the Directorate-General for Epidemiology of the Ministry of Health of the Government of Mexico.[26] The dataset consisted of over 12 million patients, each with 40 attributes. Many of the attributes, such as geographical data about the patient and health facilities, were dropped as redundant or irrelevant, retaining only those related to pre-existing medical conditions, COVID-19 positivity, and demographics. The remaining attributes were ranked according to their importance scores for each ML method, creating subsets of features with escalating cardinality to be used by different models. Data processing and predictions for each patient's outcome were achieved using models created with six different ML algorithmic methods, namely, logistic regression (LR),[27,28] decision trees (DTs),[29,30] random forest (RF),[31] eXtreme gradient boosting (XGBoost),[32] multi-layer perceptrons (MLPs),[33,34] and the k-nearest neighbors (KNN).[35] The main goal of the present study is to identify the most effective ML method for predicting COVID-19 mortality outcomes with the highest precision,[36] recall,[36] F1 score,[36] and area under the receiver-operator curve (AUC-ROC).[37]

Specifically, this study aims to:

(i). Develop ML models for COVID-19 mortality outcome prediction
(ii). Conduct a comparative analysis of COVID-19 disease mortality outcome prediction using various ML methods (LR, DTs, RF, XGBoost, MLPs, and KNN)
(iii). Evaluate the performance of different ML algorithmic methods used for the prediction of COVID-19 mortality outcome.

2. Related works

This section presents the main characteristics and outcomes of various studies conducted during the COVID-19 pandemic with the aim of predicting the mortality outcome of COVID-19 patients. The data used in these studies varied from purely clinical markers, such as blood test results, to risk factors such as heart disease, obesity, and diabetes included in the patient's history. Sample sizes varied from several hundred to millions. Similarly, the ML methods used in these studies varied, ranging from simple classifiers such as LR, DTs, and KNN to ensemble techniques such as RF, gradient boosting machine (GBM), and XGBoost.

Studies conducted at the beginning of the pandemic that used only one ML method to train the models for predicting COVID-19 patient mortality include Josephus *et al*.[38] who used the LR method in a dataset of 485 patients, and Yan *et al*.[39] who used XGBoost models with a dataset of 1,085 patients. Both studies reported an overall accuracy of 97% for their respective models. However, these studies were limited by their use of only one ML method for the model training and relatively small sample sizes.

Pourhomayoun and Shakibi[40] used a variety of ML methods, including artificial neural networks (ANNs), RF, DTs, support vector machine (SVM), KNN, and LR, to predict mortality in COVID-19 patients. Their

dataset comprised more than 2,670,000 confirmed COVID-19 patients from 146 countries, with an average age of 44.75 years. They applied feature selection to filter irrelevant symptoms and pre-existing conditions, obtaining accuracies of 89.98% for ANNs, 89.83% for KNNs, 89.02% for SVM, 87.93% for RF, 87.91% for LR, and 86.87% for DTs. The advantages of the study include the diversity the patient origins, the large sample size, and the use of different ML methods. The main limitation was the lack of ensemble ML methods.

Naseem *et al.*[41] aimed to develop a novel deep learning neural network (DNN) model for COVID-19 mortality prediction using the Neo-V framework and compared its performance to other traditional ML models, such as LR, RF, KNN, random trees, support vector classifier using radial basis function (SVC-RBF), adaptive boosting (AdaBoost) classifier, quadratic discriminant analysis, and a DNN. The dataset used comprised laboratory and clinical data of 1,214 adult COVID-19 patients admitted to Aga Khan University Hospital from February to September 2020. The DNN Neo-V model outperformed the conventional ML models, achieving an accuracy of 99.53%, a sensitivity of 89.87%, a specificity of 95.63%, and an AUC-ROC of 88.5. The main advantage of the study is the diversity of the ML methods used, including the Neo-V framework, with the limitation being the small number of patients.

Chadaga *et al.*[42] aimed to predict mortality among COVID-19 patients using epidemiological parameters. The ML methods used were RF, XGBoost, LightGBM, categorical boosting, AdaBoost, and gradient boost. The dataset used was provided by the Directorate General of Epidemiology, Secretariat of Health (Mexico)[43] and consisted of 263,007 confirmed COVID-19 patients with 19 selected attributes each. The XGBoost model achieved the best results with an accuracy of 96%, a precision of 95%, a recall of 95%, an F1-score of 95%, and an AUC-ROC of 96%. The advantages of the study include the number and variety of ML methods used and the large patient dataset. However, the main limitation was the lack of non-ensemble ML methods.

Rai *et al.*[44] proposed a voting ensemble model comprising the extra trees classifier, the RF, the gradient boosting classifier, and the XGBoost. The proposed model was compared to baseline models, including KNN, Naïve Bayes classifier, XGBoost, RF, gradient boosting classifier, and extra tree classifier. The dataset used for the research was obtained from a publicly available source consisting of blood biomarkers of 4,711 patients admitted to the hospital from March 1 to April 16, 2020. The highest scores were recorded by the proposed voting ensemble model, with an accuracy of 86.99%, a precision of 0.744, a recall of 0.690, an AUC-ROC of 0.895, and an F1-score of 0.716. The advantage of the study is the use of a diverse ensemble of ML methods, while the main limitation was the small number of patients.

Bárcenas and Fuentes-García[45] conducted a study to determine the risk factors associated with mortality in COVID-19 patients using RF, GBM, and XGBoost. They used a subset of the dataset provided by the Mexican government,[26] recorded from January 17, 2020, to June 28, 2020, which consisted of 583,678 patients, 220,657 of whom were confirmed COVID-19 patients. Patients were classified into three risk categories: low, moderate, and high, depending on comorbidities and major symptoms. The overall accuracy for predicting mortality was 89.97% for XGBoost, 89.86% for RF, and 83.37 for GBM. The advantage of the study is the large patient dataset, while its limitations include the use of only a few ML methods and the lack of non-ensemble methods. In addition, recent studies demonstrate promising results in the use of ML in various domains, such as contact tracing for COVID-19 transmission,[46] prenatal screening,[47] predicting the occurrence of type 2 diabetes,[48] and cardiovascular disease.[49]

3. Data and methods

In this section, we present the dataset used, the preprocessing techniques applied, and the ML algorithmic methods employed to train the models.

3.1. Data preprocessing

The data preprocessing procedure encompasses cleaning, transforming, and encoding the raw data, as well as generating the training and testing datasets for each iteration.

3.1.1. Cleansing

Our dataset consists of 12,425,179 cases suspected of having COVID-19, who attended various health facilities in Mexico from January 17, 2020, until January 3, 2022. The dataset is publicly available as a CSV file disseminated by the Government of Mexico.[26]

First, we translated all 40 attribute names from Spanish to English. Second, we cleansed the dataset by retaining only the positive COVID-19 cases, as indicated by the "Laboratory Result" (1: SARS-CoV-2 positive) and "Final Classification" (1, 2, and 3: Confirmed case) attributes, in accordance with the guidelines of the Epidemiological Association of Mexico and the Mexican Commission of Medical Decisions. Third, we discarded 184,345 cases containing invalid (98: Ignored and 99: Not Specified) or null values in one or more of their attributes. Thus, we

ended up with 3,809,119 COVID-19 patients with valid attribute values. Fourth, we removed the non-correlated attributes – for instance, those related to geography, residency, and indigeneity – resulting in 24 attributes. Finally, we transformed the "Date of Death" attribute into a categorical attribute by renaming it "Survived" and replacing all the "9999-99-99" date values with "1" (Yes) and the rest with "0" (No). We also combined the "Symptom Onset Date" and "Hospital Admission Date" attributes into a numerical attribute labeled "Days from Symptom to Hospitalization." Hence, we settled with 23 attributes, as shown in Table 1. Nineteen of these attributes were included in the ML models, with "Survived" being the target attribute used for predictions, and the remaining four being the "Registration ID" and three indicators. The created "gold standard dataset" (Table 2) consisting of the 23 attribute values of the 3,809,119 patients was saved in CSV file format. The dataset cleansing process flowchart is shown in Figure 1. The dataset value distribution for the categorical attributes is depicted in Figure 2, whereas Figure 3 illustrates the distribution of age groups for both genders. In Figure 3, we observe that most of the patients

Table 1. The 23 attributes of each patient

Attribute name	Values
Registration ID	Patient's unique identification code
Sex	1: Female; 2: Male
Age	Numerical positive
Smoker	1:Yes; 2:No
Pneumonia	1:Yes; 2:No
Diabetes	1:Yes; 2:No
Obesity	1:Yes; 2:No
COPD	1:Yes; 2:No
Asthma	1:Yes; 2:No
Immunosuppressed	1:Yes; 2:No
Hypertension	1:Yes; 2:No
Cardiovascular disease	1:Yes; 2:No
Chronic kidney failure	1:Yes; 2:No
Other chronic disease	1:Yes; 2:No
Pregnancy	1:Yes; 2:No; 97: Not applicable (Male)
Contact with COVID-19 case[a]	1:Yes; 2:No
Laboratory result[a]	1: SARS-CoV-2 positive; 2: SARS-CoV-2 negative; 3, 4: Not clear
Final classification[a,b]	1, 2, 3: Confirmed case; 4: Invalidly identified case; 5, 6, 7: Unconfirmed case
Patient type	1: Not admitted; 2: Admitted
Intubated	1:Yes; 2:No; 97: Not applicable
ICU	1: Admitted to ICU; 2:Not admitted to ICU; 97: Not applicable
Days from symptom to hospitalization[c]	Numerical positive (created attribute)
Survived[b]	1:Yes; 2:No (created attribute)

Notes: [a]Indicators; [b]COVID-19 sample classification; [c]Created attributes.
Abbreviations: COPD: Chronic obstructive pulmonary disease; ICU: Intensive care unit.

Table 2. The golden standard dataset table

Attribute name	Value distribution
Registration ID	3,809,119 unique values
Sex	1,921,058: Female; 1,888,061: Male
Age	Mean: 40.723; Standard deviation: 17.173; Minimum: 0; Maximum: 121
Smoker	250,558: Yes; 3,558,561: No
Pneumonia	395,528: Yes; 3,413,591: No
Diabetes	403,336: Yes; 3,405,783: No
Obesity	448,412: Yes; 3,360,707: No
COPD	31,477: Yes; 3,777,642: No
Asthma	74,682: Yes; 3,734,437: No
Immunosuppressed	23,825: Yes; 3,785,294: No
Hypertension	526,891: Yes; 3,282,228: No
Cardiovascular disease	44,362: Yes; 3,764,757: No
Chronic kidney failure	42,703: Yes; 3,766,416: No
Other chronic disease	60,307: Yes; 3,748,812: No
Pregnancy	29,332: Yes; 1,891,726: No; 1,888,061: Not applicable (Male)
Contact with COVID-19 case	1,519,968:Yes; 2,289,151: No
Laboratory result	3,802,238: SARS-CoV-2 positive; 0: SARS-CoV-2 negative; 6,881: Not clear
Final classification	3,809,119: Confirmed cases; 0: Invalidly identified cases; 0: Unconfirmed cases
Patient type	3,284,671: Not admitted; 524,448: Admitted
Intubated	60,539: Yes; 463,909: No; 3,284,671: Not applicable
ICU	42,095: Admitted to ICU; 482,353: Not admitted to ICU; 3,284,671: Not applicable
Days from symptom to hospitalization	Mean: 3.673; Standard deviation: 3.160; Minimum: -13[a]; Maximum: 43
Survived	3,558,390: Yes; 250729: No

Note: [a]The minimum value is a negative number in the cases where the patient contacted the disease inside the hospital where he was being treated.
Abbreviations: COPD: Chronic obstructive pulmonary disease; ICU: Intensive care unit.

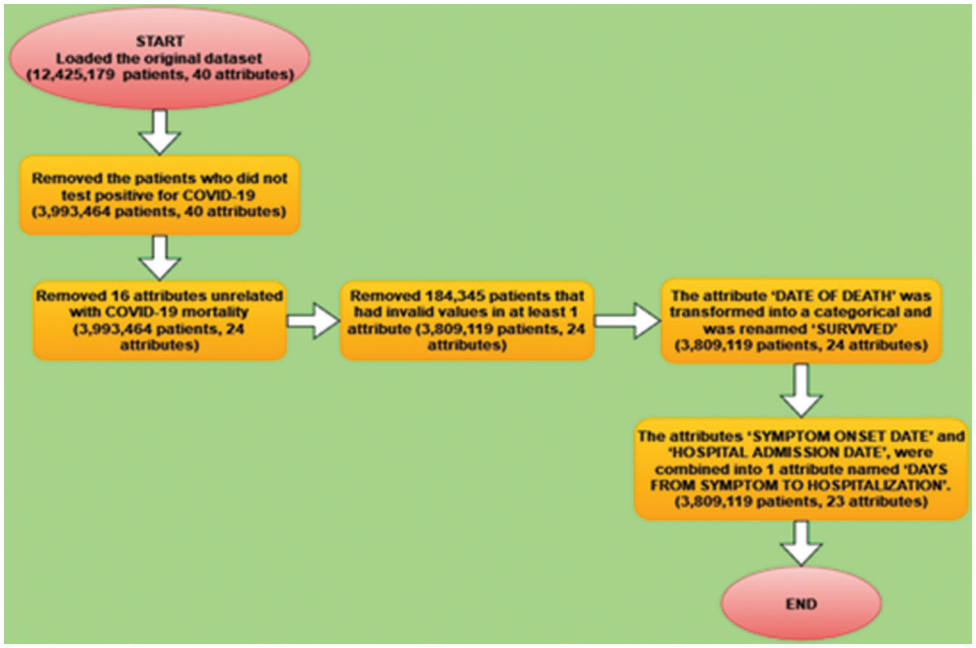

Figure 1. Data cleansing process flowchart. Image created using Draw.io (https://app.diagrams.net/)

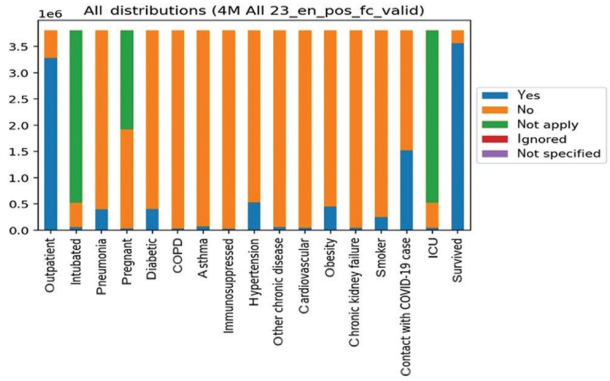

Figure 2. Distribution plot of categorical attribute values in the valid COVID-19 dataset. Image created using Python's Matplotlib library. Abbreviations: COPD: Chronic obstructive pulmonary disease; ICU: Intensive care unit.

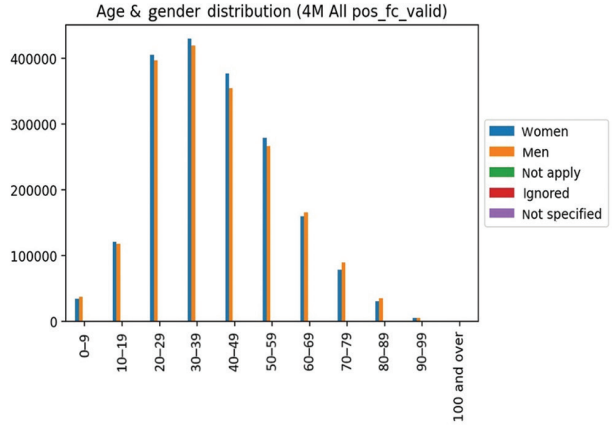

Figure 3. Age group and gender distribution chart in the valid COVID-19 dataset. Image created with Python's Matplotlib library

are between 20 and 59 years old and that the two genders are equally divided across all age groups.

3.1.2. Transformation-encoding

We further encoded and transformed the data values of the newly constructed dataset using Python's statistical analysis libraries and methods. The first step was to encode 14 out of the 17 categorical attributes, excluding "Pregnancy," "Intubated," and "ICU," using sklearn's "LabelEncoder" method. This method assigns a unique value between 0 and n-1 to each distinct value of an attribute, where n is the number of distinct values for that attribute. Next, we used the "get_dummies" method from the pandas library to convert the categorical attributes "Pregnancy," "Intubated," and "ICU" into dummy-pointer variables. These three attributes can take three distinct values: "1" for "Yes," "2" for "No," and "97" for "Not Applied." The "get_dummies" method breaks each attribute into m-1 attributes, where m is the number of distinct values for that attribute. For instance, the "Pregnancy" attribute was split into two new attributes: "Pregnancy_2" and "Pregnancy_97." Here, the values "1,0" describe a non-pregnant female, "0,1" describe a male, and "0,0" describe a pregnant female. The same transformation was applied to the "Intubated" and "ICU" attributes, resulting in "Intubated_2," "Intubated_97,"

"ICU_2," and "ICU_97." After this processing, we created three more categorical attributes, bringing the total to 20. After adding the two numerical attributes "Age" and "Days from Symptom to Hospitalization," we ultimately settled on a final total of 22 attributes used by the ML models. Finally, using the sklearn's statistical methods, namely "StandardScaler" (std) and "MinMaxScaler" (mm), we created six distinct datasets with different normalization schemes for the numerical attributes "Age" and "Days from Symptom to Hospitalization." Specifically, one dataset was created using the "StandardScaler" method, four using the "MinMaxScaler" method with different ranges (0 – 1, 0 – 10, 0 – 100, and 0 – 1000), and one without any normalizing method (none). Each of the six datasets was saved in CSV file format. The transformation-encoding process flowchart is shown in Figure 4.

3.1.3. Train-test set generation

To create the train and test sets for each ML model iteration, we formed a dataset by randomly selecting 20% of the samples from the current dataset file using the "sample" function from the pandas library. Next, to mitigate the imbalance in the "Survived" attribute, which had an approximate dead-to-survivor ratio of 1:14, we applied the synthetic minority oversampling technique (SMOTE)[50] from Python's imblearn library. This adjustment created a set with a dead-to-survivor ratio of 1:10. Finally, we applied the "RandomUnderSampler" method from the imblearn library to create the final dataset with a dead-to-survivor ratio of 1:2. This dataset was then randomly divided into two subsets: the train set, consisting of 70% of the data, and the test set, consisting 30% of the data. The flowchart illustrating the process of generating the train and test sets is depicted in Figure 5.

3.2. Models and algorithmic methods

In this study, we used six ML algorithmic methods, i.e., LR,[27,28] DTs,[29,30] RF,[31] XGBoost,[32] MLPs,[33,34] and KNN.[35] The models were implemented in Python (version 3.7) using the integrated development environment (IDE) software Spyder (version 5.1.5) and the pandas library (version 1.3.5).

3.2.1. LR

LR is a supervised learning method developed by David Cox in 1958[27] that aims to solve classification problems. LR is a generalized form of simple linear regression, used for solving classification problems where both numerical variables and categorical variables can be used as dependent variables. LR models data using the sigmoid function to make predictions about different possible outcomes.[28] Specifically, it predicts the value of dependent variables based on the weights of each independent variable. The weight of each variable is related to the degree of correlation it has with the dependent variable. In this study, we used the "LogisticRegression" method from Python's sklearn library.

3.2.2. DTs

DTs are a non-parametric supervised learning method used for classification or regression problems. The main goal of DTs is to build a model that predicts the value of a target variable by learning simple decision rules inferred from data features. The algorithm takes the given dataset and divides it into categories consisting of entities with the same value for a specific variable (attribute). This process is repeated recursively until the DT is constructed through the rules of the individual categorizations of the specific model.[30] A tree can be thought of as a piecewise consistent approximation. In this study, we constructed our DT models using the "DecisionTreeClassifier" method from Python's sklearn library.

3.2.3. RF

RF is an ensemble-supervised ML method that can be applied to both classification and regression problems. RF improves model performance by combining multiple classifiers to solve complex problems.[31] Specifically, RF is a classifier that consists of a number of DTs, each trained on a different subset of the training set. The final decision (prediction) is made by the majority vote for categorical variables or by averaging the values for numerical variables, enhancing the model's accuracy. The higher the number of DTs that comprise the forest, the higher the model's accuracy and the lower the risk of overfitting. In this study, we used the "RandomForestClassifier" method from the sklearn library.

3.2.4. XGBoost

XGBoost is a well-known variant of the gradient boosting algorithm, developed to increase prediction accuracy. XGBoost is an ensemble learning method based on DTs, utilizing a gradient-boosting framework. This framework corrects mistakes from previous DT models by modifying the weights of the variables, thereby improving subsequent models. The method was originally developed by Tianqi Chen and described by him and Carlos Guestrin in 2016.[32] XGBoost has gained widespread popularity due to its performance in ML competitions.[32] In this study, we used the "XGBClassifier" method from Python's XGBoost library.

3.2.5. MLPs

MLPs are another term for ANNs since the artificial neuron is also called a "Perceptron."[51] MLPs are a

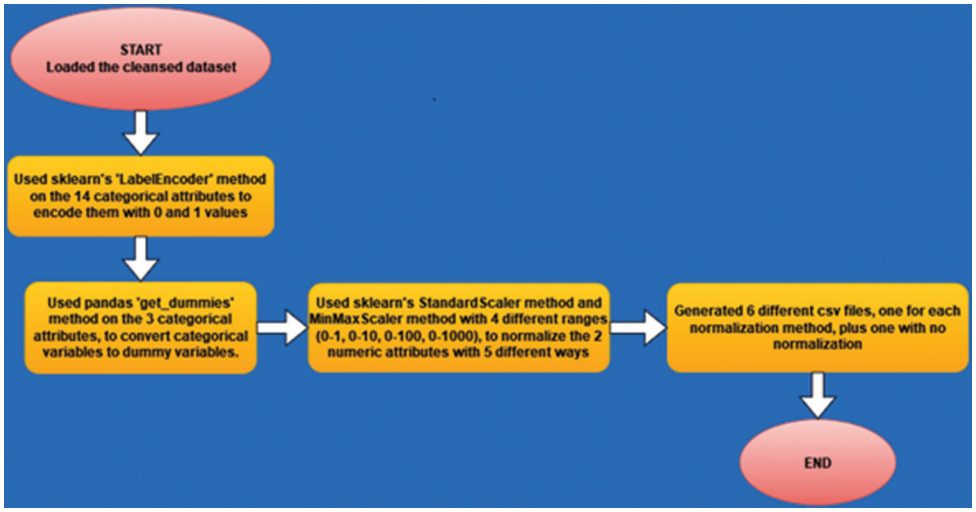

Figure 4. Data transformation-encoding process flowchart. Image created using Draw.io (https://app.diagrams.net/)

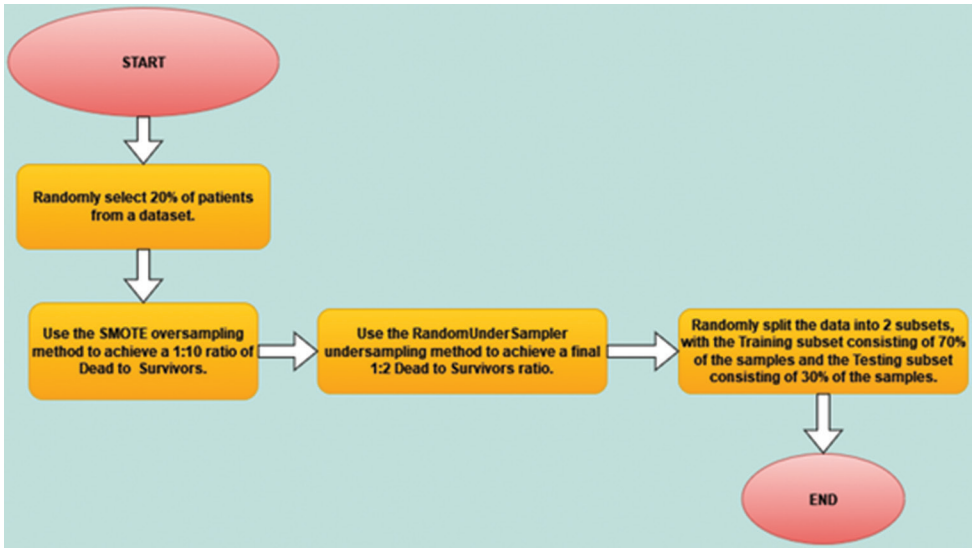

Figure 5. Train-test set generation process flowchart. Image created using Draw.io (https://app.diagrams.net/).
Abbreviation: SMOTE: Synthetic minority oversampling technique.

supervised learning method that mimics the function of neurons through algorithmic techniques to solve complex computational problems.[33,34] Given a set of attributes as independent variables and one independent variable as the target, MLPs can "learn" a non-linear approximation of the function for either classification or regression. MLPs consist of several layers of neurons: the input layer, the output layer, and the in-between or hidden layers. Each neuron is connected to every neuron of the previous and next layers. The input layer's neurons receive the data used to make predictions and pass it to the hidden layers. Finally, the output layer's neurons receive the values from the last hidden layer and make predictions for the corresponding classification or regression problem. As more layers are added to ANNs, the gradients of the loss function approach zero, making the network hard to train due to the vanishing gradient problem. This problem can be overcome through a multi-pronged approach, varying from the utilization of rectified linear unit activations to new algorithms exploring fresh techniques or enhancing existing ones.[52] MLPs are capable of learning non-linear models in real-time, but, due to the hidden layers, they exhibit a non-convex loss function owing to multiple local minima. Therefore, different random attribute weight initializations can lead to different validation accuracies, as MLPs are sensitive to the scaling of attribute weights.[53] In the present study, we used the "MLPClassifier" method from the sklearn library.

3.2.6. KNNs

The KNNs are a non-parametric supervised learning method used in classification problems, where "non-parametric" means that the input and output data will be similar in type. The method was discovered by Fix and Hodges[35] in 1951 and was subsequently developed by Cover.[54] KNN classifies new samples based on their value distance from samples with a known class label, relying on the logic that similar samples belong to the same class.[55] The class to which each new sample will belong depends on its distance from the k previous samples in the training dataset. KNN can be used for classification problems with discrete variable objectives or regression problems with continuous variable objectives. In this study, we used the "KNeighborsClassifier" method from the sklearn library.

3.3. The importance of attributes

For each ML method, except for MLPs and KNN, we used three different sets of attributes, depending on the importance score that each attribute aggregated according to the "feature_importances_" method. This sklearn library method is a vector of shape available in certain Python predictors and provides a relative measure of the importance of each feature in the predictions of the model.[56] For the "MLPClassifier" and "KNeighborsClassifier," the score for each attribute was calculated as the normalized sum of the scores from the four previous methods: "LogisticRegression," "DecisionTreeClassifier," "RandomForestClassifier," and "XGBClassifier."

These three sets had a different number of attributes: One contained all 22 attributes, another included the 15 most important attributes and the last contained only the 10 most important attributes. The following diagrams in Figures 6-16 illustrate the attribute rankings and the SHAP (SHapley Additive exPlanation) summary plots for all six ML methods.

3.4. Hyperparameter values optimization

We used three different sets of hyperparameters for each ML method. The first set contained the default values (default), the second set contained the first set of optimized values for the ML method's hyperparameters (opt_01), and the third set contained the second set of optimized hyperparameters values (opt_02). These two optimized hyperparameter sets were created using the "GridSearchCV" method from the sklearn library. To form the two sets of optimized hyperparameters, including the optimal values for most hyperparameters, we applied sklearn's "GridSearchCV()" grid search method. This method is used to search for the optimal value of each hyperparameter through a given grid containing all possible values of the different hyperparameters. In this study, we used "GridSearchCV()" as a 10-fold cross-validation method. It accepts the respective ML method and the sets of hyperparameter values as input and outputs the optimal value for each hyperparameter. This process resulted in nine different combinations for every ML method across the six datasets, creating a total of 54 different models for each ML method, and 324 models in total for all six methods. We ran each model 10 times (iterations) and calculated the mean to avoid extreme values in the metrics, resulting in 540 iterations for each ML method and a total of 3,240 iterations for all ML methods. The flowchart for creating, training, and evaluating each model is shown in Figure 17.

4. Results

This section presents the metrics used in the evaluation and the evaluation results of the created models. The evaluation results are presented for both all models created as well as the models with the highest overall score for each ML method. In addition, an overall ranking of all models according to their highest score is presented.

4.1. Evaluation metrics

To assess the performance of 324 ML models, we used the metrics of precision,[36] recall,[36] F1 score,[36] and the AUC-ROC,[37] computed through the confusion matrix,[36] and the runtime metric. The confusion matrix is a summary of the prediction results of a model, depicting the number of correct and incorrect predictions made by the evaluation model. The predictions are categorized into four groups: True positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).[36] The TP is the correctly predicted positive value, FP is the wrongly predicted positive value, TN is the correctly predicted negative value, and FN is the wrongly predicted negative value for the samples of the training set. Based on these four parameters, we can calculate precision, recall, F1 Score (F1 Score), and the AUC-ROC.

Precision is calculated as the ratio of TP to the total predicted positive observations, giving us the model's percentage of correctly predicted positive values. It is given by Equation I.

$$\text{Precision}=\frac{TP}{(TP+FP)} \quad \text{(I)}$$

Recall is the ratio of TP to the total number of positive values. It is given by Equation II below.

$$\text{Recall}=\frac{TP}{(TP+FN)} \quad \text{(II)}$$

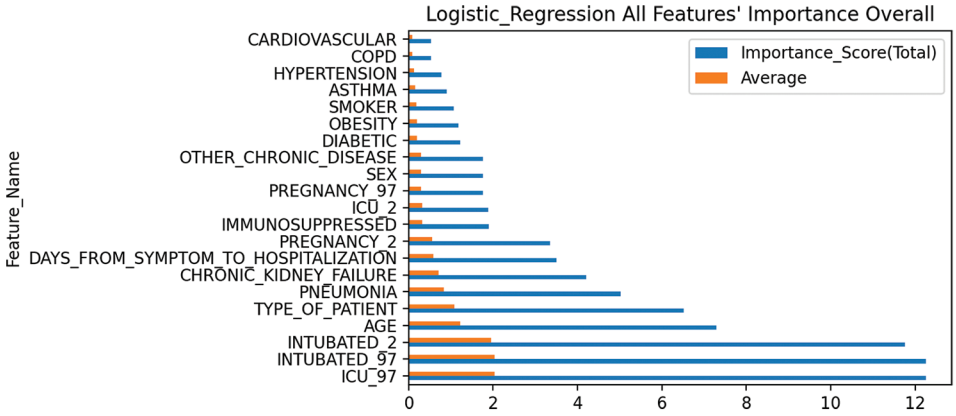

Figure 6. Attribute importance ranking of the "LogisticRegression" method. Image created using Python's Matplotlib library

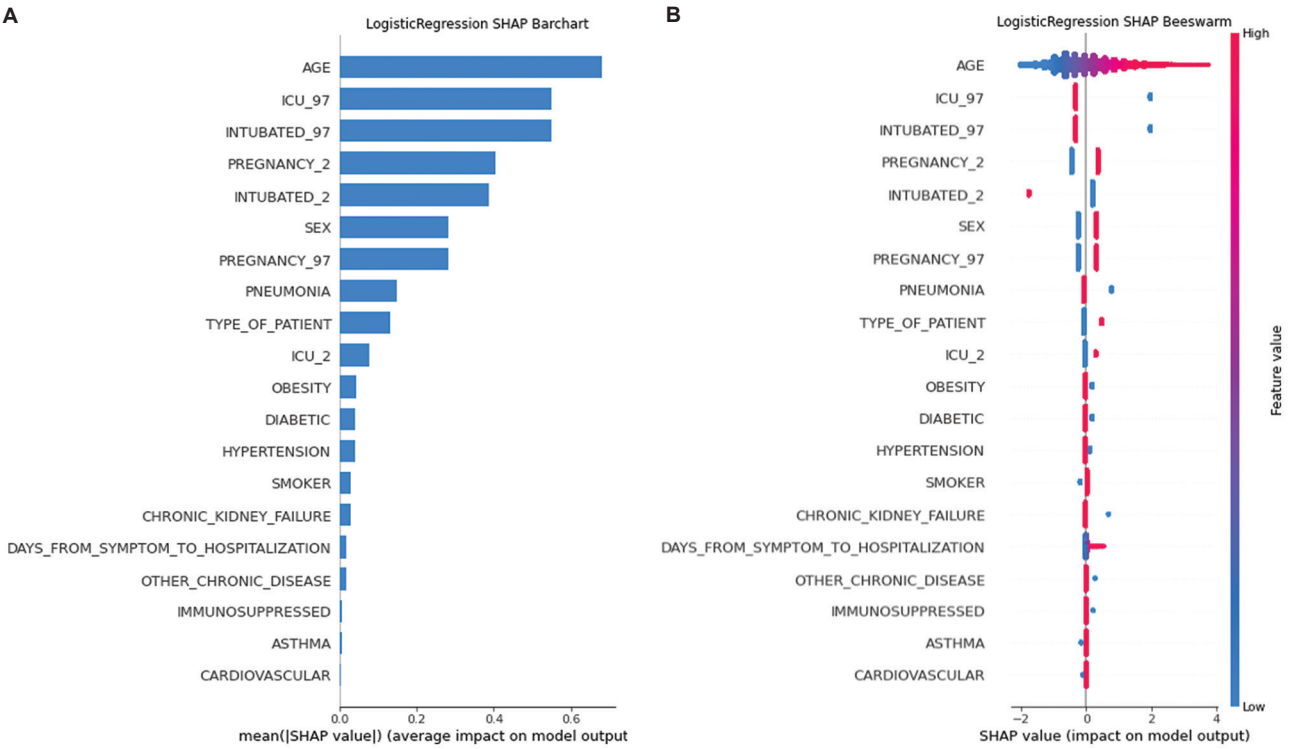

Figure 7. SHAP summary plots of the "LogisticRegression" method. (A) Barchart. (B) Beeswarm. Image created using Python's Matplotlib library

The F1 score is the weighted average of precision and recall, taking into account both FP and FN. It is usually more useful than precision, especially if there is an uneven target class distribution. The F1 score computation is given by Equation III.

$$\text{F1 score} = 2 \times \frac{\text{Recall} \times \text{Precision}}{(\text{Recall} + \text{Precision})} \quad \text{(III)}$$

The AUC-ROC is calculated as the entire two-dimensional area under the receiver operating characteristic (ROC) curve (Figure 18), from 0.0 to 1.1.[37] The ROC curve depicts the performance of the ML model being evaluated across all classification thresholds. Specifically, the ROC curve is a representation of the true positive rate (TPR) and false positive rate (FPR).[37] As the classification threshold is lowered, the model classifies more items as positive, resulting in an increase for both FPs and TPs. The value of AUC-ROC ranges from 0 to 1; for example, for a model with 100% inaccurate predictions, the AUC-ROC will be 0.00, whereas for a model with 100% accurate predictions, the AUC-ROC will be 1.00.[37] TPR and FPR are calculated using Equations IV and V, respectively.

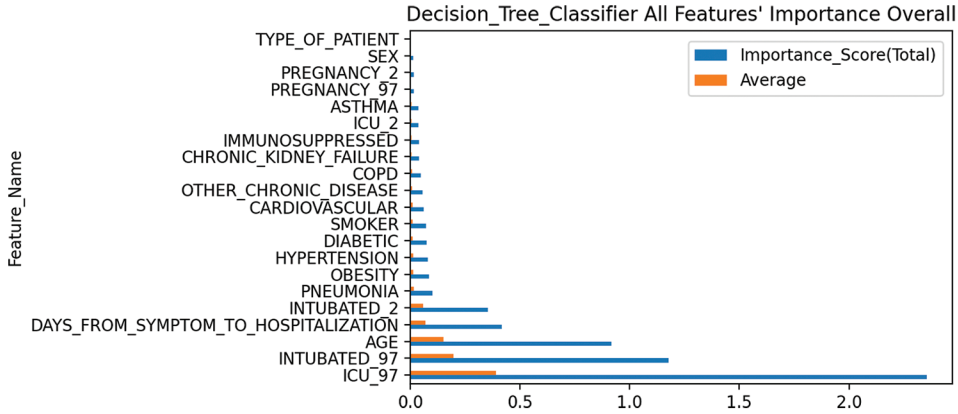

Figure 8. Attribute importance ranking of the "DecisionTreeClassifier" method. Image created using Python's Matplotlib library

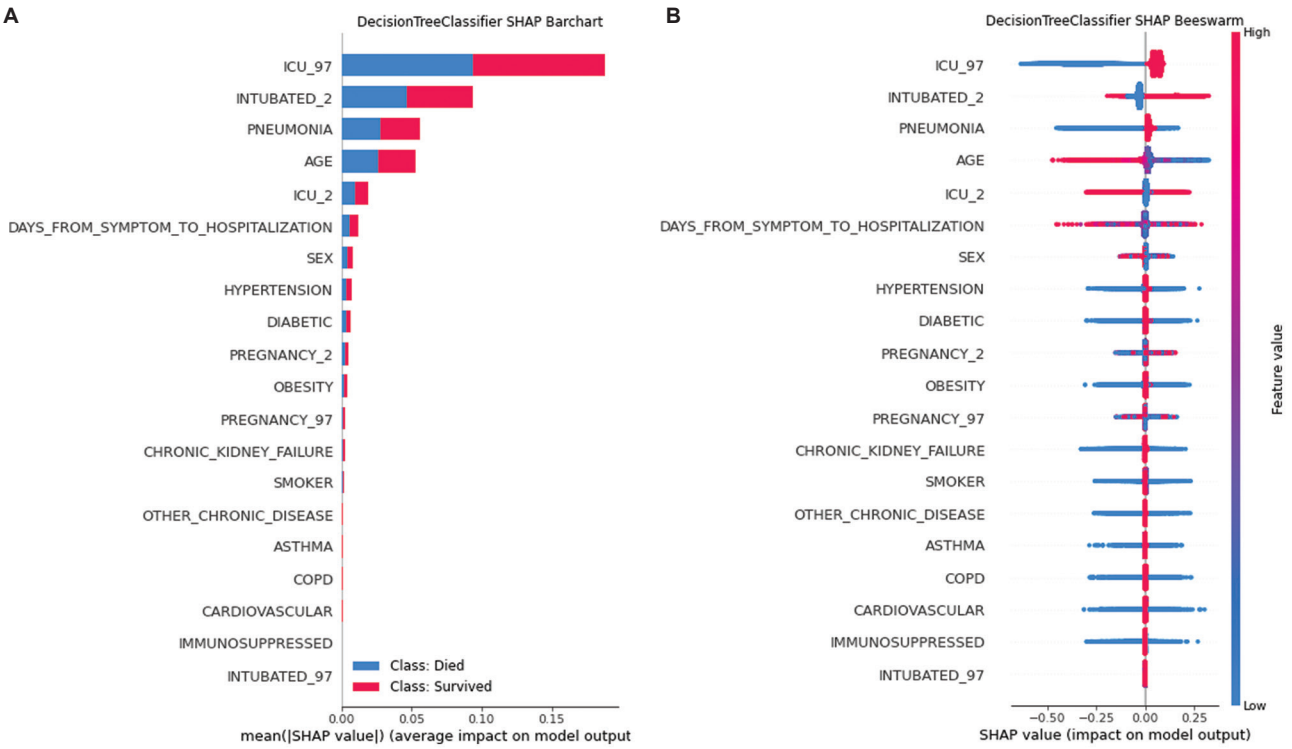

Figure 9. SHAP summary plots of the "DecisionTreeClassifier" method. (A) Barchart. (B) Beeswarm. Image created using Python's Matplotlib library

$$TPR=\frac{TP}{(TP+FN)} \quad (IV)$$

$$FPR=\frac{FP}{(FP+TN)} \quad (V)$$

Runtime is an additional metric that we used; it is measured in seconds and defined as the duration of a model's iteration.

4.2. Model evaluation

After running and evaluating all 324 different models, we ranked them according to their scores. The ML models achieved precision ranging from 90.09% to 93.76%, recall from 83.42% to 96.99%, F1-score from 84.96% to 91.13%, AUC-ROC from 0.9003 to 0.9788, and runtime from 1.092 to 910.173 s. The results of this ranking are depicted in Figure 19. The model with the highest score is positioned on the leftmost position, with the metric values decreasing as we move toward the right, so the last model scores the

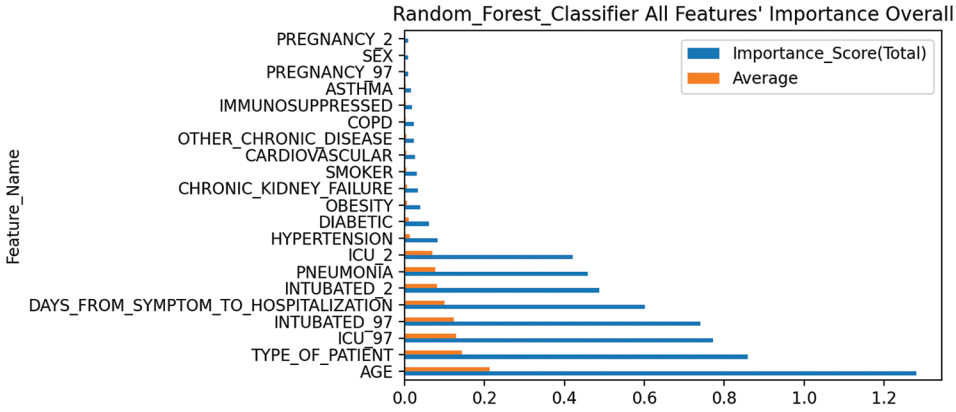

Figure 10. Attribute importance ranking of the "RandomForestClassifier" method. Image created using Python's Matplotlib library

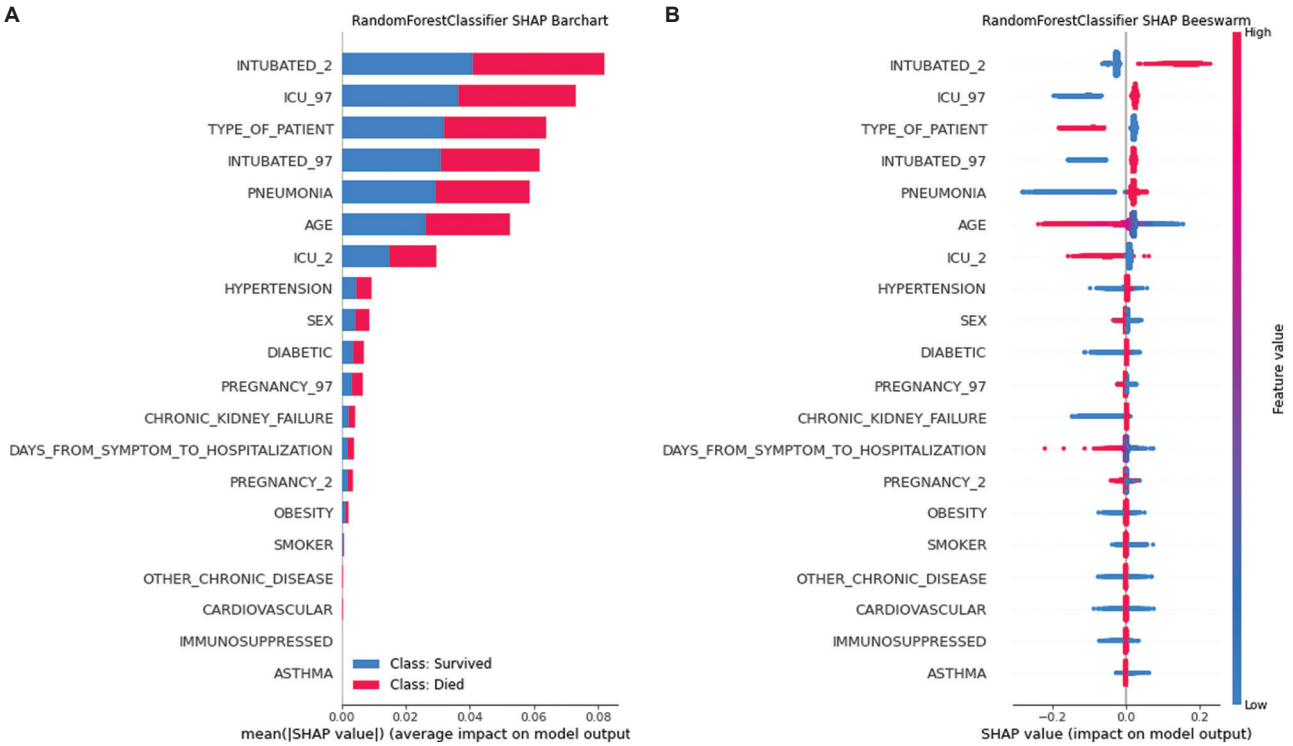

Figure 11. SHAP summary plots of the "RandomForestClassifier" method. (A) Barchart. (B) Beeswarm. Image created using Python's Matplotlib library

lowest. In the following sections, we analyze and rank the performance of the different ML methods' models both for each metric and overall for all metrics, presenting each method's highest-performing model. The ranges of the values of each metric for all models and for the three highest-ranking models for each ML method are illustrated in Tables 3 and 4, respectively. All appendix files are publicly available on GitHub (https://github.com/NikosKourb/Patients_Mortality_COVID-19_ML). All the model processes described in this paper were run in a Spyder 5.3.3 version IDE using Python 3.7.1 version as the programming language in a Microsoft Windows 10 Enterprise (x64) Build 19045.3570 (22H2) environment. The hardware used was a DELL Inspiron 3576 laptop, with an Intel(R) Core(TM) i7-8550U CPU (4 cores/8 threads/1.80GHz base/4.00GHz max), 8GB of DDR4 (2400/PC4-19200/1200.0 MHz) SDRAM, an Intel UHD Graphics 620 (Kaby Lake R U GT2) video adapter, and a DELL 0J11DH motherboard.

4.2.1. Precision

The XGBoost models showed the highest precision values, ranging from 93.21% (113th position) to 93.76%

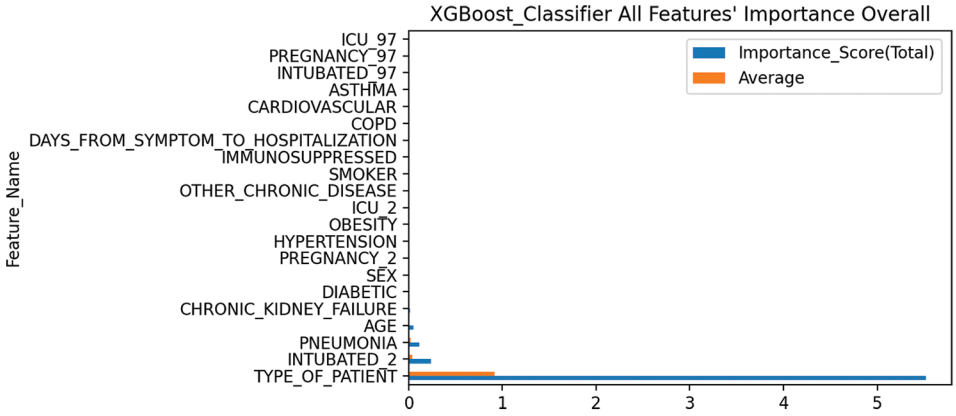

Figure 12. Attribute importance ranking of the "XGBClassifier" method. Image created using Python's Matplotlib library

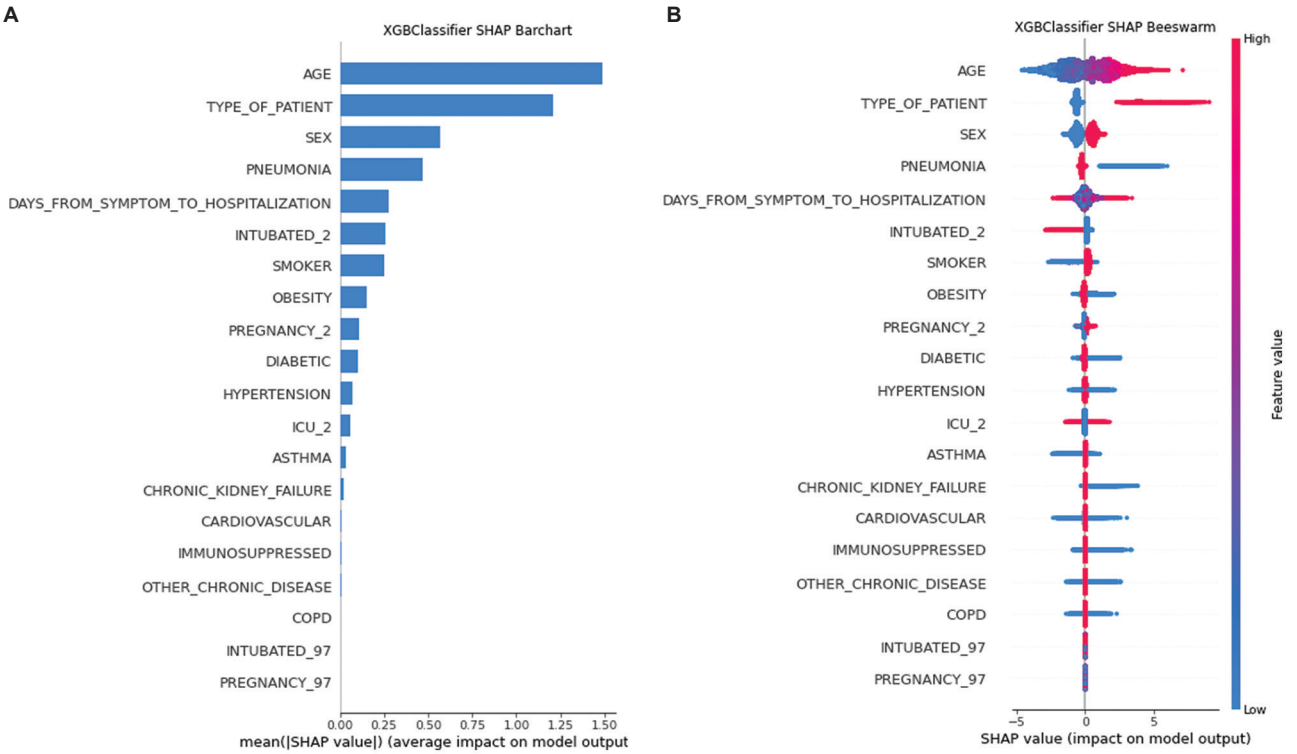

Figure 13. SHAP summary plots of the "XGBClassifier" method. (A) Barchart. (B) Beeswarm. Image created using Python's Matplotlib library

(1st position). Half of the XGBoost models (28/54) ranked above the 55th place. The highest-ranked XGBoost models processed datasets using the "Min–Max" method, with ranges of 0 – 100 and 0 – 1000 (mm_0 – 100 and mm_0 – 1000, respectively) and used all 22 attributes with the first set of optimized (opt-01) hyperparameter values.

Following XGBoost, the RF models exhibited precision values ranging from 92.25% (273rd position) to 93.66% (11th position). The distribution of RF model rankings demonstrated significant dispersion, with 50% of the models ranking above the 146th position. The highest-ranked RF models processed datasets with the "Min–Max"" method, using ranges of 0 – 100 and 0 – 1000 (mm_0 – 100 and mm_0 – 1000, respectively), and used either 22 or 15 attributes with the second set of optimized (opt-02) hyperparameter values.

The MLP models ranked third, with precision scores ranging from 92.59% (172nd position) to 93.46% (32nd position), with half of them ranking above the 88th position. The highest-ranked MLP models processed datasets with the "Min–Max" method, using ranges of

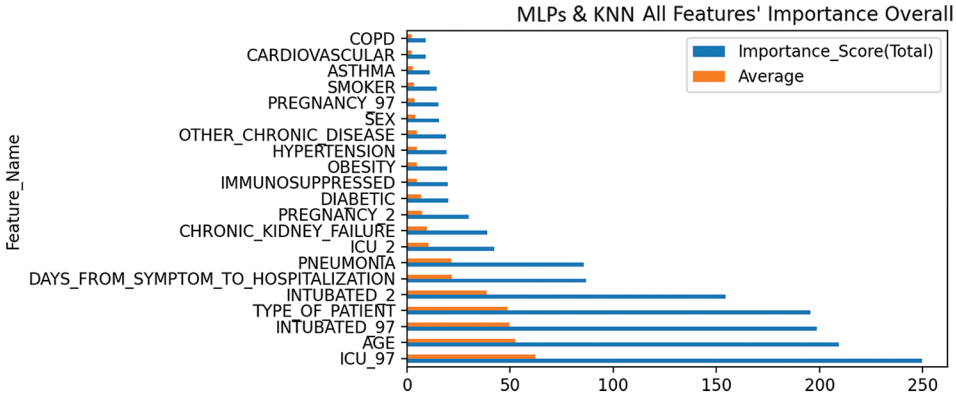

Figure 14. Attribute importance ranking of "MLPClassifier" and "KNeighborsClassifier" methods. Image created using Python's Matplotlib library

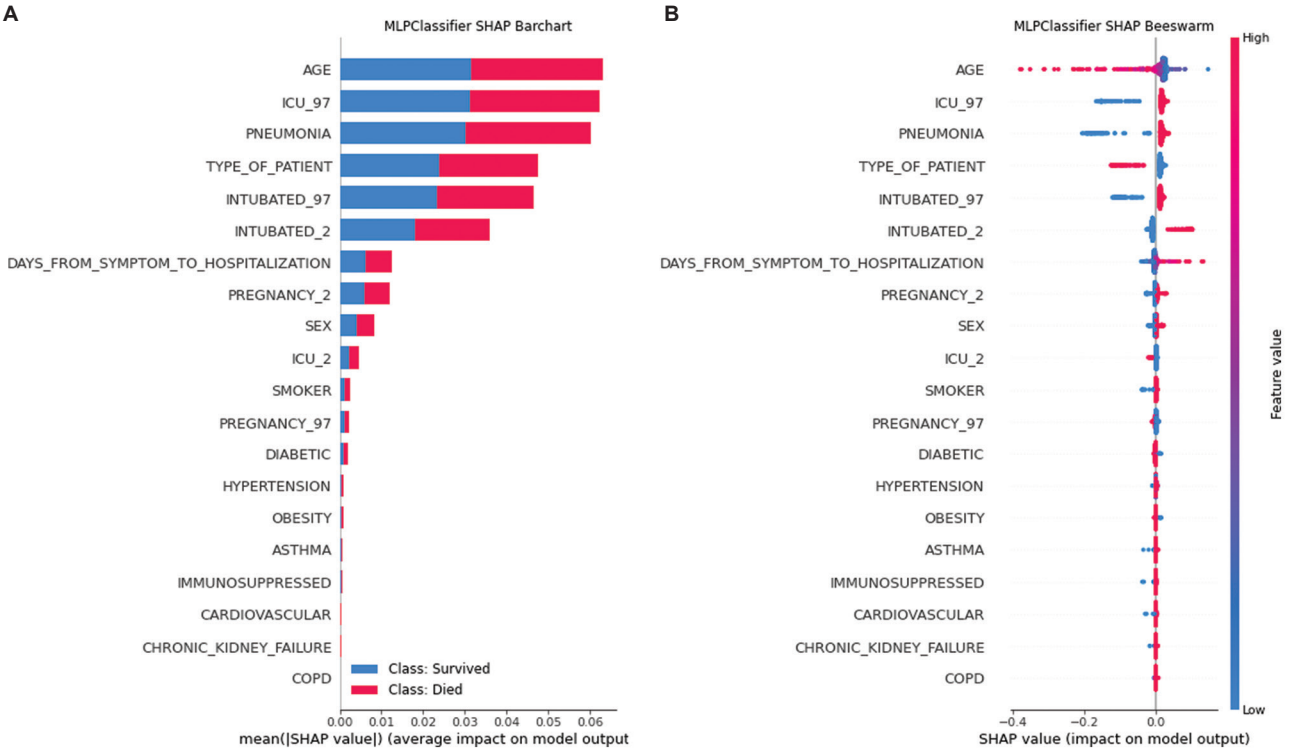

Figure 15. SHAP summary plots of the "MLPClassifier" method. (A) Barchart. (B) Beeswarm. Image created using Python's Matplotlib library

0 – 100 and 0 – 1000 (mm_0 – 100 and mm_0 – 1000, respectively), and used 22 attributes with either the first or the second sets of optimized (opt-01 and opt-02, respectively) hyperparameter values.

In the fourth place were the DT models, with precision scores ranging from 90.09% (324th position) to 93.04% (127th position). The highest-scoring DT models handled datasets that were either not processed with any normalization method (none) or processed with the "Min–Max" method, with a range of 0 – 1000 (mm_0 – 1000), and used either 22 or 15 attributes with the first set of optimized (opt-01) hyperparameter values.

K-nearest neighbor models ranked fifth, with precision scores ranging from 91.54% (304th position) to 92.85% (142nd position). The highest-scoring KNN models used datasets processed with the "StandardScaler" method (std), used either 22 or 15 attributes, and employed the first set of optimized (opt-01) hyperparameter values.

Finally, the LR models showed precision values ranging from 92.32% (267th position) to 92.63% (169th position).

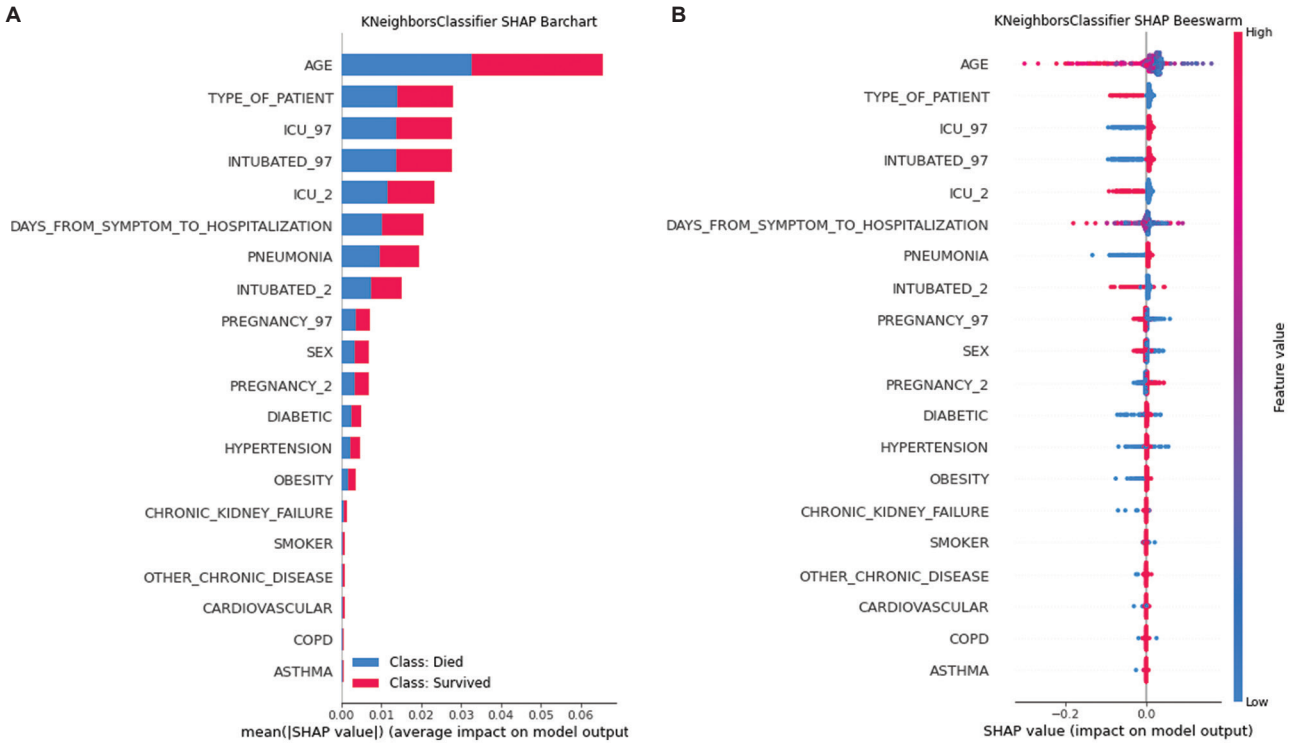

Figure 16. SHAP summary plots of the "KNeighborsClassifier" method. (A) Barchart. (B) Beeswarm. Image created using Python's Matplotlib library

The highest-ranked LR models processed datasets processed using the "Min–Max" method, with ranges of 0 – 100 and 0 – 1000 (mm_0 – 100 and mm_0 – 1000, respectively), and used either 22 or 15 attributes with the default set of (default) hyperparameter.

4.2.2. Recall

The RF models showed the highest values for recall, ranging from 90.83% (271th position) to 96.99% (1st position). The distribution of the model's rankings demonstrates significant dispersion, with 33.3% (18/54) ranking above the 19th position and 44.4% (24/54) occupying positions between 119th and 176th. The highest-ranked RF models handled datasets processed with the "Min–Max" method, using ranges of 0 – 100 and 0 – 1000 (mm_0 – 100 and mm_0 – 1000, respectively), and used either 22 or 15 attributes with the second set of optimized (opt-02) hyperparameter values.

Following the RF models, the XGBoost models ranked second, with recall values ranging from 93.96% (133rd position) to 95.61% (20th position), and 63% (34/54) ranking above the 70th position. The highest-ranked XGBoost models processed datasets using the "Min–Max" method, with ranges of 0 – 100 and 0 – 1000 (mm_0 – 100 and mm_0 – 1000, respectively), and used either 22 or 10 attributes with either the default or the first set of optimized (opt-01) hyperparameter values.

In third place were the MLPs models, scoring from 93.43% (142nd position) to 95.62% (19th position), with 88.9% (48/54) occupying positions between 51st and 113th. The highest-ranked MLPs models processed datasets using the "Min–Max" method, with ranges of 0 – 100 and 0 – 1000 (mm_0 – 100 and mm_0 – 1000, respectively) and used either 22 or 10 attributes with either the first or the second sets of optimized (opt-01 and opt-02, respectively) hyperparameter values.

In the fourth place are the DTs models, scoring from 83.42% (324th position) up to 93.69% (135th position). The highest scoring DT models handled datasets that were either not processed with any normalization method (none) or were processed with the "Min–Max" method, with 0 – 1000 (mm_0 – 1000) range, used either 22 or 15 attributes and the first set of optimal (opt-01) hyperparameter values.

In fifth place were the KNN models, scoring from 88.23% (303rd position) to 92.95% (157th position). The highest-ranked KNN models used datasets processed with the "StandardScaler" method (std) and used either 22 or 15 attributes with the first set of optimized (opt-01) hyperparameter values.

Lastly, the LR models showed recall values ranging from 90.79% (278th position) to 91.89% (182nd position). The highest-ranked LR models processed datasets using

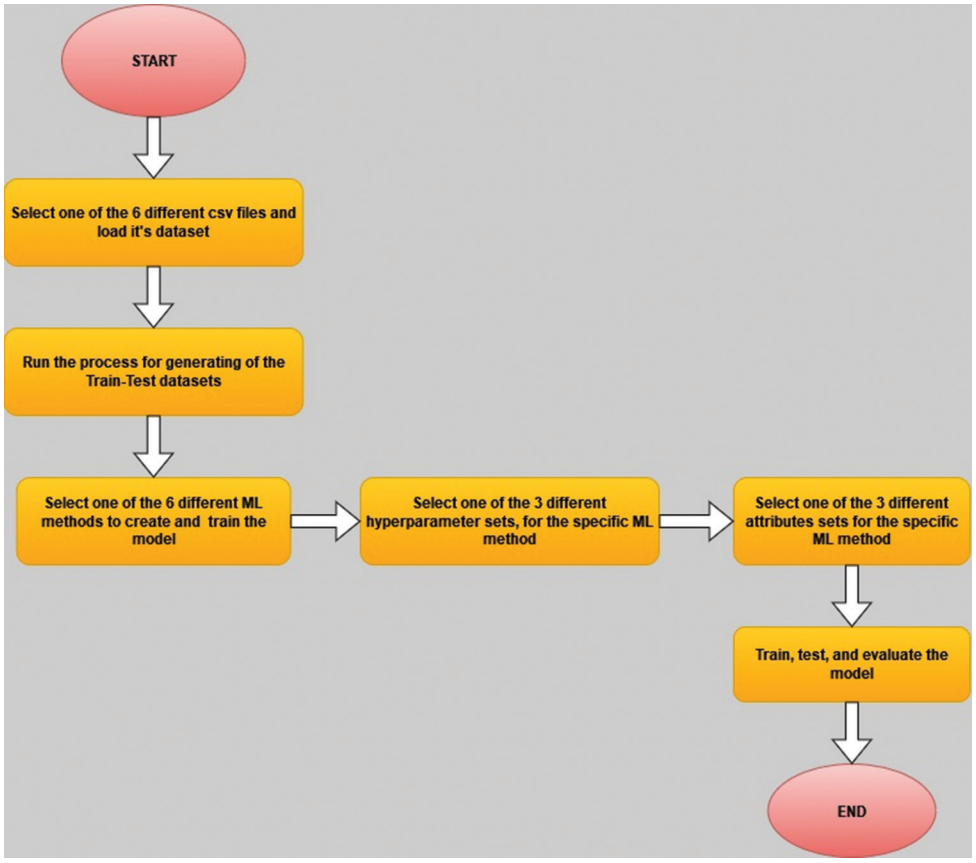

Figure 17. Creation, training, and evaluation process flowchart for each of the 324 models. Image created using Draw.io (https://app.diagrams.net/). Abbreviation: ML: Machine learning.

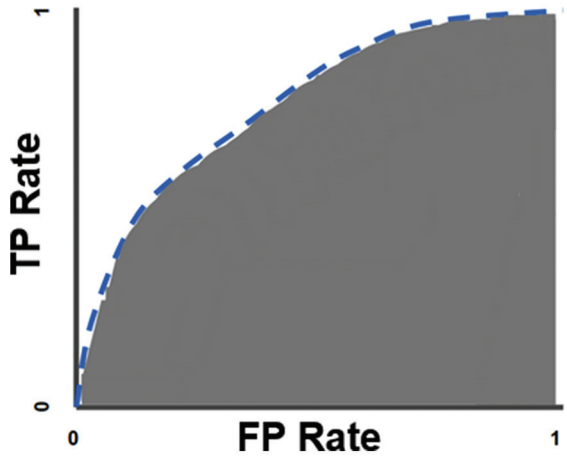

Figure 18. The area under the receiver-operating characteristic curve.[37] Abbreviations: FP: False positive; TP: True positive.

the "Min–Max" method, with ranges of 0 – 1 and 0 – 1000 (mm_0 – 1 and mm_0 – 1000, respectively) and used either 22 or 10 attributes with either the first or the second sets of optimized (opt-01 and opt-02, respectively) hyperparameter values.

4.2.3. F1 score

The XGBoost models demonstrated the highest F1 scores, ranging from 90.33% (121st position) to 91.13% (1st position), with 63% (34/54) ranking above the 54th position. The highest-ranked XGBoost models processed datasets using the "Min-Max" method, with ranges of 0 – 10, 0 – 100, and 0 – 1000 (mm_0 – 10, mm_0 – 100, and mm_0 – 1000, respectively), used 22 attributes, and employed the first set of optimized (opt-01) hyperparameter values.

The RF models secured second place, with values ranging from 88.73% (274th position) to 91.13% (2nd position). The highest-ranked RF models handled datasets processed with the "Min–Max" method, with ranges of 0 – 100 and 0 – 1000 (mm_0 – 100 and mm_0 – 1000, respectively), used either 22 or 15 attributes, and utilized either the default or the second set of optimized (opt-02) hyperparameter values.

In the third place were the MLPs models, which scored from 89.39% (168th position) to 90.76% (34th position). The highest-ranked MLP models used datasets processed

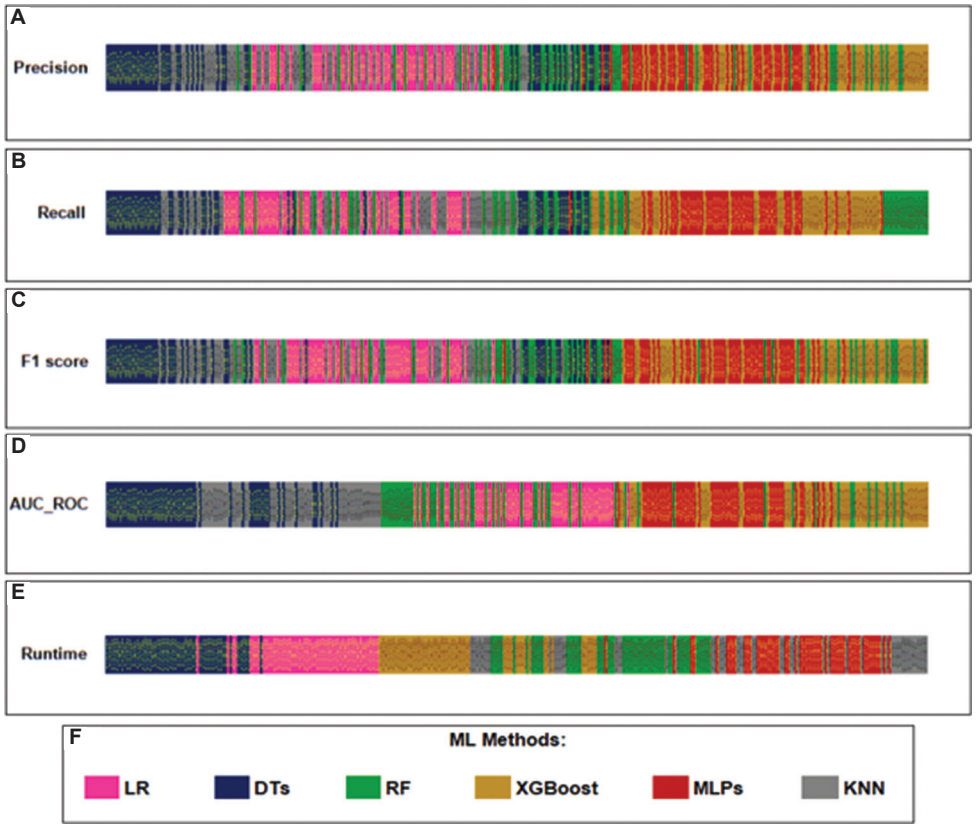

Figure 19. Ranking all 324 different models for the different metrics. (A) Ranking for precision. (B) Ranking for recall. (C) Ranking for F1 score. (D) Ranking for AUC-ROC. (E) Ranking for runtime. (F) Color-coding matching for the different ML methods. Image created using Microsoft Excel. Abbreviations: AUC-ROC: Receiver operating characteristic curve; DTs: Decision trees; KNN: K-nearest neighbors; LR: Linear regression; ML: Machine learning; MLPs: Multi-layer perceptrons; RF: Random forest; XGBoost: eXtreme gradient boosting.

with the "Min–Max" method, with ranges of 0 – 100 and 0 – 1000 (mm_0 – 100 and mm_0 – 1000, respectively), used either 22 or 15 attributes, and employed either the first or the second sets of optimized (opt-01 and opt-02, respectively) hyperparameters.

The DT models ranked fourth, scoring from 84.96% (127th position) to 90.03% (324th position). The highest-scoring DT models handled datasets that were either not processed with any normalization method (none) or were processed with the "Min–Max" method, with the 0 – 1000 (mm_0 – 1000) range, used either 22 or 15 attributes, and employed the first set of optimized (opt-01) hyperparameter values.

Following the DTs models, in fifth place, were the KNN models that scored from 87.52% (304th position) to 89.74% (145th position). The highest-scoring KNN models used datasets processed with the "StandardScaler" method (std), used either 22 or 15 attributes, and employed the first set of optimized (opt-01) hyperparameter values.

In the last place were the KNN models, with values ranging from 88.88% (266th position) to 89.26% (172nd position). The highest-ranking KNN models handled datasets processed with the "Min–Max" method, with ranges of 0 – 1 and 0 – 1000 (mm_0 – 1 and mm_0 – 1000, respectively), used all 22 attributes, and employed either the default or the second sets of optimized (opt-02) hyperparameter values.

4.2.4. Area under the receiver operating characteristic curve

The XGBoost models showed the highest values for AUC-ROC, with values ranging from 97.07% (122nd position) to 97.88% (1st position). A significant portion, 74.1% (40/54), of the XGBoost models ranked above the 58th position. The top-performing XGBoost model's processed datasets processed using the "Min–Max" method, with ranges of 0 – 100 and 0 – 1000 (mm_0 – 100 and mm_0 – 1000, respectively), used all 22 attributes, and employed either the default or the first set of optimized (opt-01) hyperparameter values.

The RF models secured second place, with AUC-ROC values ranging from 96.16% (216th position) to 97.71% (11th position). The distribution of the RF models showed a

Table 3. Performance results of all models for each ML method

ML methods	Metrics				
	Precision	Recall	F1-score	AUC-ROC	Runtime (sec)
LR	92.32 – [a]92.63%	90.79 – [a]91.98%	88.88 – [a]89.26%	0.9646 – 0.9708	1.343 – 3.317
DTs	[a]90.09 – 93.04%	[a]83.42 – 93.69%	[a]84.96 – 90.03%	[a]0.9003 – [a]0.9567	[a]1.092 – [a]1.722
RF	92.25 – 93.66%	90.83 – [b]96.99%	88.73 – 91.12%	0.9616 – 0.9771	11.951 – 30.221
XGBoost	[b]93.21 – [b]93.76%	[b]93.96 – 95.61%	[b]90.33 – [b]91.13%	[b]0.9707 – [b]0.9788	4.969 – 17.711
MLPs	92.59 – 93.46%	93.43 – 95.62%	89.39 – 90.76%	0.9706 – 0.9735	[b]18.144 – 362.466
KNN	91.54 – 92.85%	88.23 – 92.95%	87.52 – 89.74%	0.9415 – 0.9593	8.060 – [b]910.173

Notes: [a]Lowest metric values; [b]Highest metric values.
Abbreviations: AUC-ROC: Receiver operating characteristic curve; DTs: Decision trees; KNN: K-nearest neighbors; LR: Linear regression; ML: Machine learning; MLPs: Multi-layer perceptrons; RF: Random forest; XGBoost: eXtreme gradient boosting.

Table 4. Performance results of the three models, with the highest overall scores, for each ML method

ML methods	Metrics				
	Precision	Recall	F1-score	AUC-ROC	Runtime (sec)
LR	[a]92.56 – [a]92.63%	[a]90.99 – [a]91.29%	[a]89.14 – [a]89.26%	0.9704 – 0.9708	2.994 – 3.235
DTs	92.98 – 93.04%	93.37 – 93.69%	89.94 – 90.03%	[a]0.9549 – [a]0.9567	[a]1.147 – [a]1.451
RF	93.55 – 93.66%	[b]96.63 – [b]96.99%	90.96 – [b]91.13%	0.9744 – 0.9771	26.678 – 29.847
XGBoost	[b]93.73 – [b]93.76%	95.40 – 95.61%	[b]91.09 – 91.12%	[b]0.9778 – [b]0.9788	6.128 – 6.741
MLPs	93.40 – 93.46%	95.28 – 95.62%	90.67 – 90.76%	0.9726 – 0.9729	117.228 – 181.845
KNN	92.71 – 92.85%	92.56 – 92.95%	89.50 – 89.74%	0.9583 – 0.9593	[b]48.215 – [b]882.944

Notes: [a]Lowest metric values; [b]Highest metric values.
Abbreviations: AUC-ROC: Receiver operating characteristic curve; DTs: Decision trees; KNN: K-nearest neighbors; LR: Linear regression; ML: Machine learning; MLPs: Multi-layer perceptrons; RF: Random forest; XGBoost: eXtreme gradient boosting.

significant dispersion, with the lowest half ranking between the 175th and the 216th positions. The highest-ranked RF models processed datasets using the "Min–Max" method, with ranges 0 – 100 and 0 – 1000 (mm_0 – 100 and mm_0 – 1000, respectively), used either 22 or 15 attributes, and employed either the default or the second set of optimized (opt-02) hyperparameter values.

In third place were the MLPs models, with AUC-ROC values ranging from 97.06% (123rd position) to 97.35% (39th position). A majority, 85.2% (46/54), of the MLP models occupied positions between the 58th and 113th. The highest-ranked MLP models handled datasets that were either not processed with any normalization method (none) or processed with the "Min–Max" method, within the 0 – 1000 (mm_0 – 1000) range, used either 22 or 15 attributes, and applied either the first or the second sets of optimized (opt-01 and opt-02, respectively) hyperparameter values.

The LR models ranked fourth, with AUC-ROC values ranging from 96.46% (119th position) to 97.08% (203rd position). The highest-scoring LR models handled datasets that were either not processed with any normalization method (none) or processed with the "Min–Max" method, within ranges of 0 – 100 and 0 – 10 (mm_0 – 10 and mm_0 – 100, respectively), used either 22 or 15 attributes, and applied the default set of hyperparameter values.

In fifth place were the KNN models, with AUC-ROC values ranging from 94.15% (289th position) to 95.93% (217th position). The highest-scoring KNN models handled datasets that were either not processed with any normalization method (none) or processed with the "StandardScaler" method (std), used either 22 or 15 attributes, and applied either the default or the first set of optimized (opt-01) hyperparameter values.

Finally, in the last place, were the DT models, with AUC-ROC values ranging from 90.03% (324th position) to 95.67% (234th position). The highest-ranking DT models handled datasets that were either not processed with any normalization method (none) or processed using the "Min–Max" method, within the 0 – 1000 (mm_0 – 1000) range, used either 22 or 15 attributes, and applied the first sets of optimized (opt-01) hyperparameter values.

4.2.5. Runtime

The KNN models scored the highest runtime values, ranging from 8.06013 s (1st position) to 910.1731 s

(178th position). The lowest-ranking KNN models handled datasets that were either not processed with any normalization method (none) or processed with the "Min–Max" method, within the 0 – 10 (mm_0 – 10) range, using either 10 or 15 attributes and either the default or the first set of optimized (opt-01) hyperparameter values.

The MLP models scored the second-highest runtime values, ranging from 18.14386 s (125th position) to 362.46571 s (13th position). The models with the lowest runtime scores used datasets processed with the "StandardScaler" (std) or the "Min–Max" method, within the 0 – 1 (mm_0 – 1) range, used either 10 or 15 attributes, and the default set of hyperparameter values.

The third-highest runtime values were scored by the RF models, with values ranging from 11.95152 s (170th position) to 30.22087 s (85th position). The models with the lowest runtime scores handled datasets that were processed with either the "StandardScaler" (std) or the "Min–Max" method, within the 0 – 1 (mm_0 – 1) range, used the 15 most important attributes, and the default set of hyperparameters.

The XGBoost models ranked fourth in runtime, with values ranging from 4.96984 s (214th position) to 17.71116 s (179th position). The lowest-scoring XGBoost models used datasets normalized with all different methods, used either 15 or 10 attributes, and the first set of optimized (opt-01) hyperparameter values.

The LR models ranked fifth, with runtime values ranging from 1.3429 s (286th position) to 3.31659 s (215th position). The distribution of the models' ranking positions did not show significant dispersion, with 92.6% of them (50/54) ranking between the 215th and 265th positions. The highest-ranking models handled datasets that were either not processed with any normalization method (none) or processed with the "Min–Max" method, within the 0 – 10 (mm_0 – 10) range, used either 15 or 10 attributes, and the first set of optimized (opt-01) hyperparameter values.

The DT models showed the lowest runtime values, ranging from 1.09218 s (324th position) to 1.72228 s (261st position). The distribution of the models' ranking positions did not show significant dispersion. The models with the lowest runtime values handled datasets processed with the "Min–Max" method, within the 0 – 1 and 0 – 10 (mm_0 – 1 and mm_0 – 10, respectively) ranges, using 10 attributes and the second set of optimized (opt-02) hyperparameter values.

4.2.6. Overall ranking—highest scorers

Based on the overall performance of all models (Figure 19, Tables 3 and 4), the XGBoost models ranked first. The highest overall score model, "22_mm_0 – 100_opt_01," achieved 93.76% in precision, 95.47% in recall, 91.13% in F1-score, 97.86% in AUC-ROC, and a runtime of 6.67306 s. This result is somewhat expected as XGBoost is designed to be an effective and scalable method for training ML models, particularly suitable for large datasets, such as the one used in this study. XGBoost also has a strong history of achieving high-quality results in various ML competitions.[32] The success of the top XGBoost model highlights the positive impact of hyperparameter tuning, specifically the use of the first set of optimized hyperparameters and the "Min–Max" normalization method with a range of 0 to 100.

In the second position were the RF models, with the highest overall ranking model being the "22_ mm_0 – 1000_ opt_02." This model achieved 93.66% in precision, 96.99% in recall, 91.13% in F1-score, 97.71% in AUC-ROC, and a runtime of 29.84745 s. The excellent overall performance of the RF models can be attributed to the design of the RF algorithm, where each DT in the ensemble is trained on a different subset of the data, and aggregating the predictions decreases the variation of individual DTs, leading to high accuracy results. The main reason RF models scored lower than the XGBoost ones is that the RF algorithm uses a fixed set of parameters for its entire ensemble, whereas XGBoost adjusts the internal parameters of its ensemble iteratively, enabling it to handle large-scale data more effectively. The highest-scoring RF model indicates that using the second set of optimized hyperparameters and the "Min–Max" normalization method with a range of 0 – 1000 played an important role in its performance.

The MLP models ranked third, with the highest overall scoring model being the "22_ mm_0 – 1000_opt_01." This model achieved 93.46% in precision, 95.62% in recall, 90.76% in F1-score, 97.29% in AUC-ROC, and a runtime of 133.76172 s. The performance of the MLP models can be attributed to the MLP algorithm's ability to address complex non-linear problems with both small and large datasets. However, the extent to which each independent variable is affected by the dependent variable can be challenging to determine, and the performance of MLP models is heavily dependent on the quality of training, which can be time-consuming. The top-performing MLP model suggests that using the first set of optimized hyperparameters and the "Min–Max" normalization method with a range of 0 to 1000 boosted its performance.

The DT models came in fourth, with the highest overall score achieved by "22_ mm_0 – 1000_opt_01." This model achieved 93.04% in precision, 93.69% in recall, 90.03% in F1-score, 95.67% in AUC-ROC, and a runtime of 1.45128 s. The results of the DT models can be attributed to the design of the algorithm, which, while useful for

decision-related problems, is prone to overfitting, especially when interpreting each case in large-scale datasets. The highest-scoring DT model demonstrates that using the first set of optimized hyperparameters and the "Min–Max" normalization method with a range of 0 to 1000 positively impacted its performance.

The KNN models ranked fifth, with the highest scorer being "22_std_default." This model achieved 92.85% in precision, 92.95% in recall, 89.74% in F1-score, 95.93% in AUC-ROC, and a runtime of 549.69155 s. The overall performance of the KNN models can be attributed to the design of the KNN algorithm, which, while easy to implement with few hyperparameters, struggles with large datasets due to its significant computing power and data storage requirements, making it both resource-exhausting and time-consuming. The highest-scoring KNN model indicates that using the "StandardScaler" normalization method and the default hyperparameter set, instead of the optimized sets, contributed positively to its performance.

Finally, the LR models ranked sixth and last, with the highest overall scoring model being "22_ mm_0 – 1000_ default." This model achieved 92.63% in precision, 91.29% in recall, 89.26% in F1-score, 97.05% in AUC-ROC, and a runtime of 3.17691 s. The results of the LR models can be attributed to the nature of the algorithm, which, despite being easier to implement, is limited by its assumption of linearity between dependent and independent variables—a condition rarely met in real-world data. The highest-performing LR model suggests that using the default hyperparameter set and the "Min–Max" normalization method with a range of 0 – 1000 had a positive impact on its performance.

5. Discussion

In this study, data from four million COVID-19 patients were processed and used as input for each of the 324 different ML models to predict the mortality outcomes of new patients. After the evaluation, it was clear that the ML models demonstrated high performance in all metrics, with precision reaching 93.76%. The top-performing model was created using the XGBoost method, using all attributes, a "Min–Max" scaler with a range of 0 to 100, and the first set of optimized hyperparameters. After ranking all methods based on the highest overall score, the models with the best overall performance were produced by XGBoost, followed by RF, MLPs, DTs, KNN, and LR, in descending order of overall performance. This result indicates that the ensemble models of XGBoost and RF were the most successful when applied to a dataset consisting mainly of categorical attributes with only a few numerical ones. It was also observed that models using optimized sets of hyperparameters, instead of the default ones, displayed better overall performance. Furthermore, models that applied the "Min–Max" scaling with ranges between 1 – 100 and 1 – 1000 to the numerical attributes of age and days from the symptom onset to hospitalization scored higher than models using standard scaling or no scaling. In addition, models using sets of 15 or 10 attributes exhibited lower scores compared to the ones using all 22 attributes.

Moreover, while most previous studies referenced above utilized either more traditional ML models[38,40] or only ensemble methods,[42] our study used both traditional (LR, DTs, KNN, and MLPs) and ensemble (XGBoost and RF) methods to compare their performances. Other studies have also shown promising results in predicting COVID-19 mortality by using blood biomarkers alongside demographic and medical conditions, such as the studies of Nassem *et al.*[41] and Rai *et al.*[39,42,44,45] that were mentioned in the previous sections.

An important limitation of our study is that the original dataset originated from one country, Mexico. A deviation in ML models' performance would likely occur if the dataset included patients from other countries, given the differences in healthcare systems, medical care conditions, personal hygiene, etc. Another important limitation is that the original dataset consists mainly of categorical attributes. The ML models developed here would likely show different performance if trained with datasets containing different compositions, e.g., more numerical and continuous features.

6. Conclusion

The goal of this study was to create a dependable ML model that can support medical facilities and hospitals by predicting mortality outcomes in COVID-19 patients, therefore assisting in their preliminary assessment during the pandemic. We processed a dataset containing a vast number of COVID-19 patients using a large number of ML models created and trained by six ML methods, including two ensemble methods. After evaluating all models, the ML model with the highest score achieved a precision of 93.76%, a recall of 95.47%, an f1-score of 91.13%, an AUC-ROC 0.97855, and a runtime of 6.67306 s, using patients' demographics, pre-existing medical conditions, and habits. This model can help medical experts identify COVID-19 patients at high risk of death by evaluating data from questionnaires that report demographics, medical conditions, and other attributes listed in Table 1. This prioritization can ensure that the most vulnerable patients receive priority treatment during periods of overwhelming demand on the national healthcare system.

Future work could explore the possibility of developing an even higher-performance model by using an ensemble

of the top-performing models: XGBoost, RF (precision: 93.66%, recall: 96.99%, F1-score: 91.13%, AUC-ROC: 97.71%, and runtime: 29.84745 s), MLPs (precision: 93.46%, recall: 95.62%, F1-score: 90.76%, AUC-ROC: 97.29%, and runtime: 133.76172 s), and DTs (precision: 93.04%, recall: 93.69%, F1-score: 90.03%, AUC-ROC: 95.67%, and runtime: 1.45128 s). In addition, this work could be expanded to include viral diseases from previous pandemics, such as SARS, H1N1, MERS, and Ebola, in the interest of identifying the ML models with the highest overall performance.

Acknowledgments

The authors wish to acknowledge the Health Informatics Laboratory, Faculty of Nursing, National and Kapodistrian University of Athens (Athens, Greece) for providing the facilities needed for this research study.

Funding

None.

Conflict of interest

The authors declare that they have no competing interests.

Author contributions

Conceptualization: Nikolaos Kourmpanis, John Mantas
Investigation: Nikolaos Kourmpanis, Joseph Liaskos
Methodology: All authors
Writing – original draft: Nikolaos Kourmpanis
Writing – review & editing: All authors

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data

The dataset can be retrieved through a link provided by the Government of Mexico (https://www.gob.mx/salud/documentos/datos-abiertos-152127).

Further disclosure

Part of the findings has been presented in the 21st International Conference on Informatics, Management and Technology in Healthcare (ICIMTH 2023) held in Athens, Greece, from July 1 – 3, 2023.

References

1. *Coronavirus Disease (COVID-19)*. World Health Organization; 2023. Available from: https://www.who.int/news-room/questions-and-answers/item/coronavirus-disease-covid-19 [Last accessed on 2023 Dec 13].
2. *Statement on the Second Meeting of the International Health Regulations (2005) Emergency Committee Regarding the Outbreak of Novel Coronavirus (2019-nCoV)*. World Health Organization; 2020. Available from: https://www.who.int/news/item/30-01-2020-statement-on-the-second-meeting-of-the-international-health-regulations-(2005)-emergency-committee-regarding-the-outbreak-of-novel-coronavirus-(2019-ncov) [Last accessed on 2023 Dec 18].
3. *WHO Director-General's Opening Remarks at the Media Briefing on COVID-19*. World Health Organization; 2020. Available from: https://www.who.int/director-general/speeches/detail/who-director-general-s-opening-remarks-at-the-media-briefing-on-covid-19---11-march-2020 [Last accessed on 2023 Dec [18].
4. *WHO Coronavirus (COVID-19) Dashboard*. *WHO Coronavirus (COVID-19) Dashboard With Vaccination Data*. World Health Organization; 2023. Available from: https://covid19.who.int [Last accessed on 2023 Dec 18].
5. *Naming the Coronavirus Disease (COVID-19) and the Virus that Causes it*. World Health Organization; 2023. Available from: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/technical-guidance/naming-the-coronavirus-disease-(covid-2019)-and-the-virus-that-causes-it [Last accessed on 2023 Dec 18].
6. Machhi J, Herskovitz J, Senan AM, *et al*. The natural history, pathobiology, and clinical manifestations of SARS-CoV-2 infections. *J Neuroimmune Pharmacol*. 2020;15(3):359-386.
doi: 10.1007/s11481-020-09944-5
7. Zhou P, Yang XL, Wang XG, *et al*. Addendum: A pneumonia outbreak associated with a new coronavirus of probable bat origin. *Nature*. 2020;588(7836):E6.
doi: 10.1038/s41586-020-2951-z
8. Hoffmann M, Kleine-Weber H, Schroeder S, *et al*. SARS-CoV-2 cell entry depends on ACE2 and TMPRSS2 and is blocked by a clinically proven protease inhibitor. *Cell*. 2020;181(2):271-280.e8.
doi: 10.1016/j.cell.2020.02.052
9. Lednicky JA, Tagliamonte MS, White SK, *et al*. Independent infections of porcine deltacoronavirus among Haitian children. *Nature*. 2021;600(7887):133-137.
doi: 10.1038/s41586-021-04111-z
10. Vlasova AN, Diaz A, Damtie D, *et al*. Novel canine coronavirus isolated from a hospitalized patient with pneumonia in east Malaysia. *Clin Infect Dis*. 2022;74(3):446-454.
doi: 10.1093/cid/ciab456
11. Lytras S, Hughes J, Martin D, *et al*. Exploring the natural origins of SARS-CoV-2 in the light of recombination. *Genome Biol Evol*. 2022;14(2):evac018.
doi: 10.1093/gbe/evac018

12. Zhou H, Ji J, Chen X, *et al*. Identification of novel bat coronaviruses sheds light on the evolutionary origins of SARS-CoV-2 and related viruses. *Cell*. 2021;184(17):4380-4391.e14.

doi: 10.1016/j.cell.2021.06.008

13. Wacharapluesadee S, Tan CW, Maneeorn P, *et al*. Evidence for SARS-CoV-2 related coronaviruses circulating in bats and pangolins in Southeast Asia. *Nat Commun*. 2021;12(1):972.

doi: 10.1038/s41467-021-21240-1

14. Mitchell TM. Machine Learning. McGraw-Hill Science/Engineering/Math; 1997. Available from: https://www.cin.ufpe.br/~cavmj/Machine%20-%20Learning%20-%20Tom%20Mitchell.pdf [Last accessed on 2023 Dec 18].

15. Zoabi Y, Deri-Rozov S, Shomron N. Machine learning-based prediction of COVID-19 diagnosis based on symptoms. *NPJ Digit Med*. 2021;4(1):3.

doi: 10.1038/s41746-020-00372-6

16. Aljameel SS, Khan IU, Aslam N, Aljabri M, Alsulmi ES. Machine learning-based model to predict the disease severity and outcome in COVID-19 patients. *Sci Program*. 2021;2021:1-10.

doi: 10.1155/2021/5587188

17. Mullick B, Magar R, Jhunjhunwala A, Barati Farimani A. Understanding mutation hotspots for the SARS-CoV-2 spike protein using Shannon Entropy and K-means clustering. *Comput Biol Med*. 2021;138:104915.

doi: 10.1016/j.compbiomed.2021.104915

18. Ozger ZB, Cihan P. A novel ensemble fuzzy classification model in SARS-CoV-2 B-cell epitope identification for development of protein-based vaccine. *Appl Soft Comput*. 2022;116:108280.

doi: 10.1016/j.asoc.2021.108280

19. *People with Certain Medical Conditions*. Centers for Disease Control and Prevention; 2023. Available from: https://www.cdc.gov/coronavirus/2019-ncov/need-extra-precautions/people-with-medical-conditions.html [Last accessed on 2023 Dec 18].

20. Yang X, Yu Y, Xu J, *et al*. Clinical course and outcomes of critically ill patients with SARS-CoV-2 pneumonia in Wuhan, China: A single-centered, retrospective, observational study. *Lancet Respir Med*. 2020;8(5):475-481.

doi: 10.1016/S2213-2600(20)30079-5

21. Guan WJ, Ni ZY, Hu Y, *et al*. Clinical characteristics of coronavirus disease 2019 in China. *N Engl J Med*. 2020;382(18):1708-1720.

doi: 10.1056/NEJMoa2002032

22. Cakir Edis E. Chronic pulmonary diseases and COVID-19. *Turk Thorac J*. 2020;21(5):345-349.

doi: 10.5152/TurkThoracJ.2020.20091

23. Goumenou M, Sarigiannis D, Tsatsakis A, *et al*. COVID19 in Northern Italy: An integrative overview of factors possibly influencing the sharp increase of the outbreak (Review). *Mol Med Rep*. 2020;22:20-32.

doi: 10.3892/mmr.2020.11079

24. Brake SJ, Barnsley K, Lu W, McAlinden KD, Eapen MS, Sohal SS. smoking upregulates angiotensin-converting enzyme-2 receptor: A potential adhesion site for novel coronavirus SARS-CoV-2 (Covid-19). *J Clin Med*. 2020;9(3):841.

doi: 10.3390/jcm9030841

25. Lewis T. *Smoking or Vaping May Increase the Risk of a Severe Coronavirus Infection. Scientific American*; 2020. Available from: https://www.scientificamerican.com/article/smoking-or-vaping-may-increase-the-risk-of-a-severe-coronavirus-infection1 [Last accessed on 2023 Dec 18].

26. *Datos Abiertos Dirección General de Epidemiología*. Secretaría de Salud. Gobierno. Cobierno de Mexico; 2023. Available from: https://www.gob.mx/salud/documentos/datos-abiertos-152127 [Last accessed on 2023 Dec 18].

27. Cramer JS. The origins of logistic regression. *SSRN Electron J*. 2005.

doi: 10.2139/ssrn.360300

28. *Logistic Regression in Machine Learning - Javatpoint*; 2021. Available from: https://www.javatpoint.com/logistic-regression-in-machine-learning [Last accessed on 2023 Dec 18].

29. Utgoff PE. Incremental induction of decision trees. *Mach Learn*. 1989;4(2):161-186.

30. Kotsiantis S. Decision trees: A recent overview. *Artif Intell Rev*. 2013;39(4):261-283.

doi: 10.1007/s10462-011-9272-4

31. *Machine Learning Random Forest Algorithm - Javatpoint*; 2021. https://www.javatpoint.com/machine-learning-random-forest-algorithm [Last accessed on 2023 Dec 18].

32. Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM; 2016. p. 785-794.

doi: 10.1145/2939672.2939785

33. Beale R, Jackson T. *Neural Computing: An Introduction*. England: Adam Hilger; 1990.

doi: 10.1887/0852742622

34. Bezdek JC. On the relationship between neural networks, pattern recognition and intelligence. *Int J Approx Reason*. 1992;6(2):85-107.

doi: 10.1016/0888-613X(92)90013-P

35. Fix E, Hodges JL. Discriminatory analysis. Nonparametric discrimination: Consistency properties. *Int Stat Rev/Rev Int Stat*. 1989;57(3):238.

doi: 10.2307/1403797

36. Fitton D. *Evaluating Models in Azure Machine Learning (Part 1: Classification)*. Adatis; 2020. Available from: https://adatis.co.uk/evaluating-models-in-azure-machine-learning-part-1-classification [Last accessed on 2023 Dec 18].

37. *Classification: ROC Curve and AUC. Machine Learning. Google for Developers. Google Machine Learning Education*; 2022. Available from: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc [Last accessed on 2023 Dec 18]

38. Josephus BO, Nawir AH, Wijaya E, Moniaga JV, Ohyver M. Predict mortality in patients infected with COVID-19 virus based on observed characteristics of the patient using logistic regression. *Procedia Comput Sci*. 2021;179:871-877.

doi: 10.1016/j.procs.2021.01.076

39. Yan L, Zhang HT, Goncalves J, *et al*. An interpretable mortality prediction model for COVID-19 patients. *Nat Mach Intell*. 2020;2(5):283-288.

doi: 10.1038/s42256-020-0180-7

40. Pourhomayoun M, Shakibi M. Predicting mortality risk in patients with COVID-19 using machine learning to help medical decision-making. *Smart Health*. 2021;20:100178.

doi: 10.1016/j.smhl.2020.100178

41. Naseem M, Arshad H, Hashmi SA, Irfan F, Ahmed FS. Predicting mortality in SARS-COV-2 (COVID-19) positive patients in the inpatient setting using a novel deep neural network. *Int J Med Inform*. 2021;154:104556.

doi: 10.1016/j.ijmedinf.2021.104556

42. Chadaga K, Prabhu S, Umakanth S, *et al*. COVID-19 mortality prediction among patients using epidemiological parameters: An ensemble machine learning approach. *Eng Sci*. 2021;16:221-33.

doi: 10.30919/es8d579

43. Franklin MR. *Mexico COVID-19 Clinical Data*; 2019. Available from: https://www.kaggle.com/datasets/marianarfranklin/mexico-covid19-clinical-data [Last accessed on 2023 Dec 18].

44. Rai N, Kaushik N, Kumar D, Raj C, Ali A. Mortality prediction of COVID-19 patients using soft voting classifier. *Int J Cogn Comput Eng*. 2022;3:172-179.

doi: 10.1016/j.ijcce.2022.09.001

45. Bárcenas R, Fuentes-García R. Risk assessment in COVID-19 patients: A multiclass classification approach. *Inform Med Unlocked*. 2022;32:101023.

doi: 10.1016/j.imu.2022.101023

46. Al-Shaikh A, Mahafzah BA, Alshraideh M. Hybrid harmony search algorithm for social network contact tracing of COVID-19. *Soft Comput*. 2023;27(6):3343-3365.

doi: 10.1007/s00500-021-05948-2

47. Mandala SK. Unveiling the unborn: Advancing fetal health classification through machine learning. *Artif Intell Health*. 2023;1(1):2121.

doi: 10.36922/aih.2121

48. Al-Tawil M, Mahafzah BA, Al Tawil A, Aljarah I. Bio-inspired machine learning approach to type 2 diabetes detection. *Symmetry* (*Basel*). 2023;15(3):764.

doi: 10.3390/sym15030764

49. Umar BU, Ajao LA, Dogo EM, Ajao FJ, Atama M. Artificial intelligence model for prediction of cardiovascular disease: An empirical study. *Artif Intell Health*. 2023;1(1):1746.

doi: 10.36922/aih.1746

50. Chawla NV, Bowyer KW, Hall LO, Philip Kegelmeyer W. SMOTE: Synthetic minority over-sampling technique. *J Artif Intell Res*. 2002;30(2):321-357.

51. Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol Rev*. 1958;65(6):386-408.

doi: 10.1037/h0042519

52. Abuqaddom I, Mahafzah BA, Faris H. Oriented stochastic loss descent algorithm to train very deep multi-layer neural networks without vanishing gradients. *Knowl Based Syst*. 2021;230:107391.

doi: 10.1016/j.knosys.2021.107391

53. Neural *Network Models (supervised)*; 2021. Available from: https://scikit-learn.org/stable/modules/neural_networks_supervised.html [Last accessed on 2023 Dec 18].

54. Cover TM, Hart PE. Nearest neighbor pattern classification. *IEEE Trans Inf Theory*. 1967;13(1):21-27.

doi: 10.1109/TIT.1967.1053964

55. Kubat M. *An Introduction to Machine Learning*. Berlin: Springer; 2017.

doi: 10.1007/978-3-319-63913-0

56. *Glossary of Common Terms and API*; 2007. Available from: https://scikit-learn.org/stable/glossary.html#term-feature_importances [Last accessed on 2023 Dec 18].