applied sciences

MDPI

Article

Adaptive Stacking Ensemble Techniques for Early Severity Classification of COVID-19 Patients

Gun-Woo Kim [1], Chan-Yang Ju [2], Hyeri Seok [3] and Dong-Ho Lee [2,*]

[1] Department of Computer Science and Engineering, Gyeongsang National University, Jinju 52828, Republic of Korea; gunwoo.kim@gnu.ac.kr
[2] Department of Applied Artificial Intelligence, Hanyang University, Ansan 15588, Republic of Korea; karunogi@hanyang.ac.kr
[3] Division of Infectious Disease, Department of Internal Medicine, Korea University College of Medicine, Korea University Ansan Hospital, Ansan 15355, Republic of Korea; hyeri.seok@gmail.com
* Correspondence: dhlee72@hanyang.ac.kr

Abstract: During outbreaks of infectious diseases, such as COVID-19, it is critical to rapidly determine treatment priorities and identify patients requiring hospitalization based on clinical severity. Although various machine learning models have been developed to predict COVID-19 severity, most have limitations, such as small dataset sizes, the limited availability of clinical variables, or a constrained classification of severity levels by a single classifier. In this paper, we propose an adaptive stacking ensemble technique that identifies various COVID-19 patient severity levels and separates them into three formats: Type 1 (low or high severity), Type 2 (mild, severe, critical), and Type 3 (asymptomatic, mild, moderate, severe, fatal). To enhance the model's generalizability, we utilized a nationwide dataset from the South Korean government, comprising data from 5644 patients across over 100 hospitals. To address the limited availability of clinical variables, our technique employs data-driven strategies and a proposed feature selection method. This ensures the availability of clinical variables across diverse hospital environments. To construct optimal stacking ensemble models, our technique adaptively selects candidate base classifiers by analyzing the correlation between their predicted outcomes and performance. It then automatically determines the optimal multi-layer combination of base and meta-classifiers using a greedy search algorithm. To further improve the performance, we applied various techniques, including imputation of missing values and oversampling. The experimental results demonstrate that our stacking ensemble models significantly outperform existing single classifiers and AutoML approaches, with improvements of 6.42% and 8.86% in F1 and AUC scores for Type 1, 9.59% and 6.68% for Type 2, and 11.94% and 9.24% for Type 3, respectively. Consequently, our approach improves the prediction of COVID-19 severity levels and potentially assists frontline healthcare providers in making informed decisions.

Keywords: COVID-19; early triage; clinical servility prediction; adaptive stacking ensemble

Citation: Kim, G.-W.; Ju, C.-Y.; Seok, H.; Lee, D.-H. Adaptive Stacking Ensemble Techniques for Early Severity Classification of COVID-19 Patients. *Appl. Sci.* **2024**, *14*, 2715. https://doi.org/10.3390/app14072715

Academic Editor: Alexander N. Pisarchik

Received: 22 January 2024
Revised: 12 March 2024
Accepted: 22 March 2024
Published: 24 March 2024

1. Introduction

The novel coronavirus disease (COVID-19) emerged in late 2019 and rapidly escalated into a global pandemic with profound effects on public health, economic stability, and the global social structure. In the early stages of the pandemic, some countries required hospitalization for all confirmed COVID-19 patients, regardless of the severity of their illness [1]. This policy significantly burdened medical facilities by increasing the number of patient admissions. Furthermore, the rapid increase in confirmed cases and the associated risk of mortality exacerbated the shortage of healthcare professionals and facilities. Consequently, many patients did not receive appropriate treatment, leading to symptom worsening and, in some instances, death [2].

To effectively deliver the healthcare process with limited resources, it is important to predict the clinical severity of COVID-19 cases to quickly prioritize treatment and

identify patients requiring hospitalization. COVID-19 treatment relies on symptomatic relief, supportive care, oxygen therapy, and intensive care, depending on the disease severity [3]. Therefore, it is essential to appropriately allocate the limited healthcare resources, such as quarantine facilities, hospital beds, and intensive care units, based on the different severity level classifications of COVID-19 patients.

Several machine learning model approaches have been proposed to predict the severity of COVID-19 patients. However, these studies have the following limitations.

Small patient datasets: Many studies relied on limited information or clinical variables, such as medical images [4], blood or urine tests [5,6], clinical characteristics [7], and Electronic Health Records (EHRs) [8] obtained during hospitalization. These data sources have the advantage of not requiring additional expenditure on materials, as the necessary information is already included in the patient's medical records. However, these studies typically derived their datasets from the patient pool at a single institution in a specific local region, usually involving 300–600 patients. Thus, these studies have limited generalizability due to their relatively small sample sizes.

Limited availability of clinical variables: Some studies have shown promising results using patient data obtained from extensive medical testing devices, such as blood tests [9], CT scans [10,11], and MRIs [12]. However, the time required to confirm the test results can delay the rapid identification of patients with critically severe COVID-19. In real-world clinical settings, it is often necessary to triage and refer COVID-19 patients immediately after diagnosis, even with limited available clinical variables [13]. Therefore, an adaptive approach that can predict severity using readily available clinical variables or by selecting variables that significantly impact actual clinical severity is needed.

Constrained classification of severity levels by a single classifier: Most studies have employed a single classifier, such as Logistic Regression or XGBoost, to predict COVID-19 severity [14–16]. While these models have shown reasonable predictive capabilities, their performance can decrease with insufficient data or when the number of severity levels increases, making decision boundaries more complex. As a result, some studies have focused on simple binary classifications, categorizing patient cases as either low or high severity, with an emphasis on identifying mortality risk [17–19]. Moreover, single classifiers, which utilize different learning methods, are limited in their ability to capture complex patterns or the diversity within a dataset, potentially resulting in failure to generalize effectively across the entire dataset. Consequently, each single classifier has different performance levels, and important clinical variables for severity prediction appear inconsistently across different classifiers. Furthermore, actual COVID-19 patient data, which were not collected for research purposes, present challenging issues, such as irregular sampling and data imbalance. Irregular sampling complicates data extraction and contributes to the large amounts of missing data, while data imbalances can cause classifiers to be biased towards the majority class, leading to unsatisfactory results. In this context, the performance of single classifiers can significantly vary, limiting their ability to create stable, reliable prediction models and often resulting in biased predictions towards the majority class without effectively learning from the minority class.

To address these issues, we propose an adaptive stacking ensemble technique to effectively identify various severity levels of COVID-19 patients. The main contributions of this paper are as follows.

Use of a large patient dataset for generalization: In contrast to previous methods developed with limited patient datasets from a single institution in a specific local region, we utilized a nationwide clinical epidemiology dataset that includes data from 5644 confirmed COVID-19 patients. Our dataset, collected by the South Korean government, particularly the Korea Centers for Disease Control and Prevention (KCDC), comprises patient data from over 100 hospitals nationwide. To validate our approach with an actual patient dataset, we enriched our dataset by incorporating additional data from a single institution, Korea University Ansan Hospital. Consequently, our approach not only improves the

performance of our model but also enhances its generalizability, making it applicable in diverse clinical settings.

Adaptive data strategies for clinical variable availability: The adaptive characteristics of our technique allow for the classification of COVID-19 severity using only the available clinical variables, even when certain variables are unavailable or time-consuming to collect. In contrast to previous methods that depend on extensive medical testing devices, we utilize clinical variables that can be easily obtained through patient interviews or health applications to support early patient triage. Recognizing that even easily obtainable clinical variables can vary depending on the hospital's situation, we categorize the variables within the dataset into five groups: patient demographics, vital signs, symptoms, underlying conditions, and blood test results. We employ a data-driven strategy to group these variables and develop adaptive models for their various combinations, evaluating each model's performance. Additionally, we enhance our clinical severity prediction models by using only the selected features identified through our feature selection method, considering the clinical settings that can complicate data acquisition.

Adaptive stacking ensemble models for identifying various severity levels: To overcome the limitations of single classifiers in predicting highly diverse severity levels, we propose an adaptive stacking ensemble technique that automatically combines multiple single classifiers. Our technique leverages the strengths of each classifier to capture diverse severity patterns and correlations within our dataset, thereby improving prediction performance. To construct the optimal stacking ensemble model, our technique adaptively selects base classifiers with low complexity and high diversity based on the correlation between their predicted outcomes and the performance of the candidate classifiers. We then employ a greedy search algorithm to automatically determine the optimal multi-layer combination of base and meta-classifiers. As a result, our technique represents a novel prediction approach that reduces the risk of misclassification and model bias by automatically combining selected single classifiers into a multilayer stacking ensemble learning model, after evaluating the impact of each classifier. To further improve the model's performance, we utilize missing-value imputation and oversampling techniques. These methods address the inherent challenges of clinical data that were not collected for research purposes, such as data loss due to irregular sampling and imbalanced datasets, which can degrade the performance of predictive models.

The remainder of this paper is organized as follows: Section 2 provides a literature review of related work. Section 3 presents an overview and the detailed methods of the proposed technique. Section 4 provides a comprehensive experimental evaluation and discusses our findings. Finally, we conclude the paper in Section 5.

2. Related Work

The rapid spread of COVID-19 has emphasized the need for effective early-stage severity prediction models to optimize resource allocation and patient triage. Consequently, numerous studies have proposed the development of machine learning models to predict COVID-19 severity in patients and improve prognoses.

Yan et al. [20], Shang et al. [21], and Zhang et al. [22] utilized datasets from local hospitals in Wuhan, China, during the early stages of the COVID-19 outbreak. They predicted characteristics and risk factors related to the clinical severity and mortality of COVID-19 patients using single classifiers, including fuzzy logic and multivariate Logistic Regression. Their goal was to identify clinical variables that could significantly influence COVID-19 severity based on patients' clinical characteristics and in-hospital record information (e.g., patient demographics, blood test results, medical history, and real-time PCR). The epidemiological and clinical characteristics of the patients in these datasets were identified as key factors related to prognosis heterogeneity after COVID-19 diagnosis [23]. However, the generalizability and applicability of these methods are limited due to their reliance on small patient datasets, with sizes of 375, 443, and 663 patients, respectively.

Liang et al. [24] aimed to enhance their prediction model's generalizability by using a dataset of 1590 patients from hospitals in the Wuhan, Hubei, and Guangdong regions of China. They proposed a deep-learning-based Cox proportional hazards model to predict COVID-19 patients' survival rates. However, this model requires the time-series-based tracking of patient cohort data, which can be challenging and costly to obtain due to the need for continuous data formatting.

A machine-learning-based analysis of CT images has been proposed to predict COVID-19 patient mortality rates and severity changes over time. Jin et al. [25] used a deep-learning-based diagnostic model with chest CT imaging for the early detection, quantification, and tracking of COVID-19. They integrated a large-scale dataset that included CT scans of 11,356 patients from three medical centers in China and four publicly available databases. Similarly, Xu et al. [26] explored the performance of single classifiers in rapidly triaging COVID-19 patients by combining CT image analysis with EHRs and clinical laboratory results. Al Rahhal et al. [27] utilized a Vision Transformer (ViT) technique, known for its high performance in computer vision, to enhance accuracy by segmenting and processing the training data along with augmented CT images. Despite the advantages of these studies in tracking severity changes over time using image data, the need for medical testing devices to acquire training image data poses a significant challenge, especially where rapid initial classification is required. Furthermore, Guan et al. [28] reported that approximately 20% of COVID-19 patients do not show significant imaging changes in the lungs, leading to unnecessary radiation exposure and the potential misallocation of limited testing resources.

Wungu et al. [29] used a meta-analysis to investigate the correlation between various cardiac markers and the severity or mortality of COVID-19 patients. Their findings suggested that elevated levels of CK-MB, PCT, NT-proBNP, BNP, and D-dimer could serve as indicators for predicting COVID-19 severity. Meanwhile, Bayat et al. [30] applied pairwise correlation to compress a dataset composed of 70 clinical characteristics and used the XGBoost model for prediction. They concluded that biomarkers such as ferritin, CRP, LDH, and D-dimer could potentially act as indicators for detecting COVID-19 infection. However, these datasets are obtained through standard laboratory tests, which are time-consuming and expensive. Since these tests are typically designed to diagnose specific health conditions, they cannot fully represent all potential clinical scenarios, necessitating additional tests or specialized procedures. Therefore, in the context of COVID-19 diagnosis, the availability of relevant clinical variables can be challenging due to the high influx of patients in emergency rooms and the need for rapid diagnosis in pandemic situations.

Fan et al. [31] categorized various machine-learning-based COVID-19 prediction models into Knowledge-Driven (KD) infectious disease dynamics models and Data-Driven (DD) machine learning models. KD models, derived from all available domain knowledge, incorporate known relationships of infectious disease models, while DD models rely solely on given datasets without domain knowledge. KD models typically use a set of mathematical equations, like ordinary differential equations, including physically interpretable parameters to reveal the key characteristics and transmission rules of infectious diseases. However, they may suffer from poor predictive power due to the simplified or ambiguous explanations of mechanistic processes and parameter uncertainty. DD models employ various machine learning techniques, such as Support Vector Machines (SVM), Artificial Neural Networks (ANN), Random Forests (RF), Decision Trees (DT), and K-Nearest Neighbors (KNN), to predict COVID-19 mortality risk. These models can handle unknown non-linear relationships in infectious disease transmission mechanisms through data learning but risk reflecting model bias in their predictions due to the bias–variance tradeoff.

Ensemble learning, which combines the predictive abilities of two or more base learner models to reduce bias and variance, thereby improving overall prediction performance, has seen recent application in analyzing COVID-19 data. However, determining the optimal combination of models for optimal performance remains challenging, with most researchers manually constructing stacking ensemble models to predict COVID-19's clinical severity [32–35]. Additionally, some studies opted for simplified severity classifications

related only to survival and death, rather than attempting to classify a diverse range of clinical severities [36,37].

To address the challenge of automating machine learning model construction, current Automated Machine Learning (AutoML) technology automatically generates prediction pipelines in a data-driven manner, producing high-quality predictions. AutoML reduces repetitive tasks to the data analysis and enables the development of applications without the need for specific expertise in machine learning or statistics. For example, Ikemura et al. [38] used the open-source H2O.ai AutoML package to predict COVID-19 patient mortality, and de Holanda et al. [39] utilized the PyCaret library to select algorithms for identifying COVID-19 patients at risk of death or needing hospitalization. However, most existing AutoML research focuses on finding the optimal model among single classifiers or is limited by the use of a restricted range of single classifiers, failing to ensure model diversity and often not optimizing data and feature preprocessors, focusing solely on model optimization. Although some AutoML approaches include stacking ensemble techniques, they generally have a simple single-layer structure, which limits the integration needed to address complex data structures and diverse prediction requirements.

The recent advancement of Large Language Models (LLMs) has introduced new possibilities for processing biomedical data through Natural Language Processing (NLP), including identifying COVID-19 infections. López-Úbeda et al. [40] proposed a system that automatically predicts whether patients' chest CT scan reports match radiological findings indicative of COVID-19, using BiLSTM-, CNN-, and ANN-based text classification techniques. Mermin-Bunnell et al. [41] developed an NLP model that accurately classifies EHRs and identifies COVID-19 cases using BioClinicalBERT and DistilBERT, potentially reducing clinician response time and improving access to antiviral treatments. Muzhe et al. [42] developed a method using social media data from Reddit to train a novel QuadArm model based on BERT to answer questions, aiming to identify COVID-19 cases and automatically extract reported symptoms. These NLP-based approaches typically identify patient severity by searching for similar results (e.g., words or documents) from text-based unstructured data. However, these approaches face significant challenges due to the peculiarities of medical terminology. Medical terms can be ambiguous, appearing in various forms, such as synonyms, homonyms, and abbreviations, which can vary in meaning depending on the context [43,44]. For example, the term '*cold*' can refer both to a common viral respiratory infection and a general feeling of low temperature, leading to the potential misinterpretation of patient symptoms. Similarly, the abbreviation '*MI*' might be used to denote '*myocardial infarction*' (i.e., a heart attack) in one context, while in another, it could stand for '*mitral insufficiency*' (i.e., a valve disorder). Moreover, the presence of spelling errors, grammatical inaccuracies, and colloquial abbreviations can hinder the models' ability to accurately identify patient severity or lead to incorrect word extractions.

To overcome these drawbacks and present a generalized model for predicting COVID-19 clinical severity, we performed severity prediction using a large patient dataset obtained from the KCDC and Korea University Ansan Hospital. To tackle the challenge of collecting clinical variables and implementing early triage, we utilized the epidemiological variables of patients based on categorical and numerical values, such as body mass index (BMI), body temperature, heart rate, blood pressure, and existing underlying diseases. These variables are readily available and can be quickly obtained at the time of patient admission. Moreover, considering the potential limitations of collecting patient data in real-world clinical settings, we grouped the clinical variables using data-driven strategies. We then evaluated the performance of various models using different combinations of these variable groups. We also analyzed important clinical variables using feature selection methods based on wrapper methods, including forward selection, backward elimination, and recursive feature elimination with cross-validation (RFECV). To improve performance, we addressed the issues of irregular sampling and data imbalance through missing value imputation and oversampling techniques. Finally, we presented a flexible machine learning model capable of accurately predicting the clinical severity of COVID-19 patients across various severity

levels by employing an adaptive stacking ensemble technique with multi-layer structures, which helps reduce the risk of model bias.

3. Proposed Method

This section provides a detailed description of our proposed techniques. Figure 1 presents a schematic illustration of our method, which consists of five phases: (1) preprocessing, (2) data-driven strategy, (3) feature selection, (4) data splitting and oversampling, and (5) stacking ensemble model construction. A detailed explanation of each phase is provided in the following subsections.

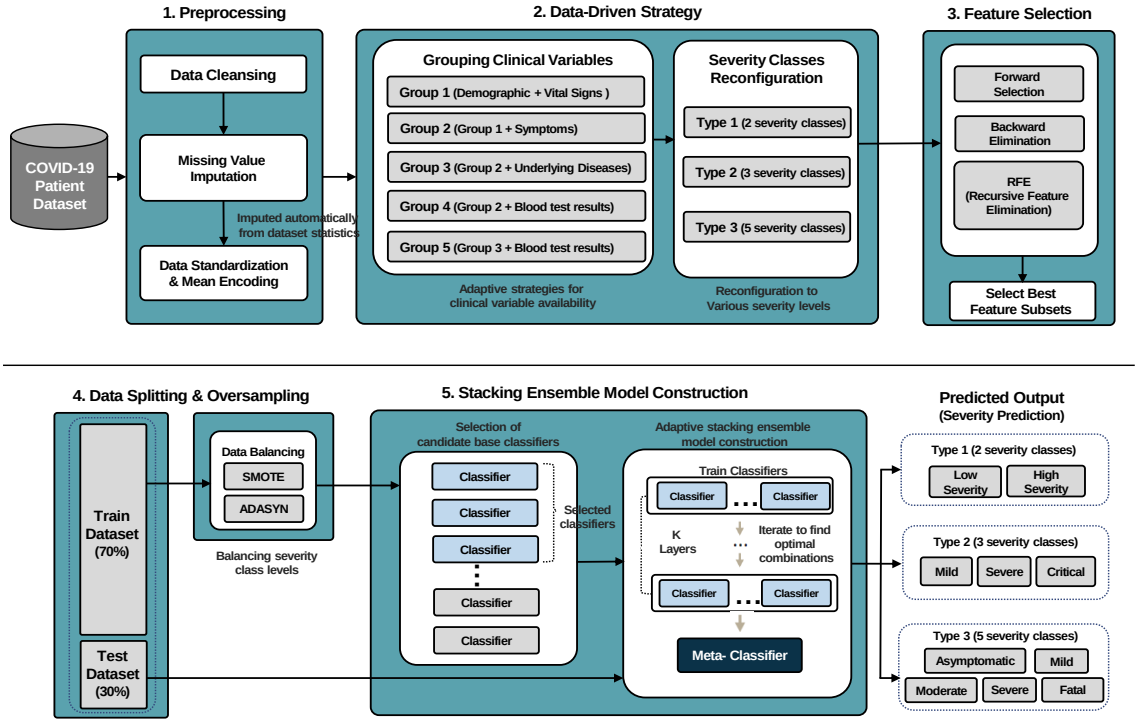


Figure 1. Overview of the proposed technique.

Our technique is specifically designed to predict the various severity levels of COVID-19 in patients. Differentiating the severity levels is essential to provide appropriate interventions, save lives, and efficiently allocate limited medical resources. Therefore, our technique can deliver the final severity prediction results, derived from the input data, in one of three output format types depending on the needs of the user's intervention: Type 1 (low or high severity), Type 2 (mild, severe, and critical), and Type 3 (asymptomatic, mild, moderate, severe, and fatal). The prediction models delivering these results are adaptively constructed based on the availability of clinical variables and built with various forms of internal stacking ensembles. The following steps summarize our technique:

(1) In the first internal phase, we apply data preprocessing techniques to the acquired COVID-19 patient data. This phase includes the removal of irrelevant features, the imputation of missing values, data standardization, and the mean encoding of categorical features.

(2) In the data-driven strategy, we create groups of clinical variables for severity classification, considering the availability of a patient's clinical variables. These groups mainly consist of epidemiological variables that can be easily obtained when the patient visits the hospital. To optimize the predictions using the minimum amount of necessary clinical variables, we modify the composition of each group by changing the feature sets (i.e., the clinical variables).

(3) To identify the feature sets that significantly affect the severity prediction, we propose a feature selection algorithm. This algorithm considers forward selection, backward elimination, and RFECV to identify the optimal feature set.
(4) Subsequently, we split the dataset into training and testing subsets. The training data undergo an oversampling process to ensure a balanced data distribution.
(5) In the stacking ensemble model construction phase, single models are built and evaluated using the F1 and AUC score metrics. To construct an adaptive stacking ensemble model, we select k single models that demonstrate excellent performance and ensure diversity, with low correlations among the predictions. The predictions are then performed using adaptive stacking ensemble models, and the results are presented in one of three format types for various severity classes.

3.1. Datasets

In this paper, we utilized a dataset comprising 5644 COVID-19 patient records. This dataset included 5628 patient records collected by the KCDC from more than 100 hospitals nationwide from 20 January to 30 April 2020. Additionally, we incorporated 50 patient records obtained from Korea University Ansan Hospital until 30 August 2020. This time period is critically important as it covers the initial outbreak, acute phase, and peak of COVID-19 cases in South Korea, providing a comprehensive overview of the virus's impact. Specifically, this period highlights the urgency of medical responses focused on efficiently managing medical processes with limited resources and rapidly determining treatment priorities.

The dataset contains 42 clinical variables for patients confirmed positive through real-time PCR testing, categorized into eight target classes that reflect the severity of each patient's condition. The clinical variables are defined as easily measurable features, such as body temperature, blood pressure, symptoms, and past medical history, which can be promptly gathered directly from patients without significant delay.

Despite its nationwide scope, the dataset does not specify the locations of diagnoses in the Republic of Korea. It includes the records of patients who were treated and subsequently discharged from quarantine or hospitalization, as well as those who died from COVID-19 complications. The discharge criteria for patients included receiving two consecutive negative test results at least 24 h apart and the absence of symptoms. To analyze different combinations of variable groups, we classified the clinical variables into five categories: patient demographics, vital signs, symptoms, underlying diseases, and blood test results.

3.2. Preprocessing

To effectively utilize the COVID-19 patient dataset, this phase involves several preprocessing steps, including data cleansing, missing value imputation, data standardization, and mean encoding. Initially, we conducted data cleansing to remove irrelevant features. We excluded variables that did not contribute to the prediction process, such as *PatientID*, *Outcome*, and *Duration*. Additionally, we excluded the variable *InpatientRoom* because of the potential bias caused by incorrect patient placement. Thus, we utilized 38 of the 42 clinical variables as the model inputs. Detailed descriptions of the selected clinical variables are presented in Table 1. The target variable for prediction, *Clinical Severity Score (CSS)*, was missing in 27 of 5678 records. Furthermore, we discovered seven patients in the dataset who died before their COVID-19 status was definitively confirmed through real-time PCR, despite undergoing laboratory tests for COVID-19. These records were excluded, and 5644 patient records were used to develop our clinical severity prediction model.

Table 1. Clinical variables of COVID-19 patient dataset.

Clinical Variables	Type	Components	Patient (n = 5644)	Missing Values
Patient Demographics				
AGE (Years)	9 categories	(1) 0–9 years (2) 10–19 years (3) 20–29 years (4) 30–39 years (5) 40–49 years (6) 50–59 years (7) 60–69 years (8) 70–79 years (9) Over 80 years	66 (1.17%) 205 (3.63%) 1110 (19.67%) 566 (10.03%) 740 (13.11%) 1149 (20.36%) 920 (16.30%) 554 (9.82%) 334 (5.92%)	0 (0%)
SEX (Gender)	2 categories	(1) Male (2) Female	2333 (41.34%) 3311 (58.66%)	0 (0%)
PREG (Pregnancy)	2 categories	(1) Yes (2) No	19 (0.34%) 3276 (58.04%)	2349 (41.62%)
PREGW (Pregnancy Week)	5 categories	(1) 0 weeks (2) 1–9 weeks (3) 10–19 weeks (4) 20–29 weeks (5) Over 30 weeks	50 (0.89%) 26 (0.46%) 8 (0.14%) 4 (0.07%) 2 (0.04%)	5554 (98.41%)
BMI (Body Mass Index)	4 categories	(1) Less than 18.5 (underweight) (2) 18.5–24.9 (normal) (3) 25.0–29.9 (overweight) (4) Greater than 30 (obese)	260 (4.61%) 2919 (51.72%) 1061 (18.80%) 210 (3.72%)	1194 (21.16%)
Vital Signs				
SBP (Systolic Blood Pressure)	5 categories	(1) Less than 120 (2) 120–129 (3) 130–139 (4) 140–159 (5) Greater than 160	1317 (23.33%) 1145 (20.29%) 1092 (19.35%) 1433 (25.39%) 522 (9.25%)	135 (2.39%)
DBP (Diastolic Blood Pressure)	5 categories	(1) Less than 80 (2) 80–89 (3) 90–99 (4) Greater than 100	2131 (37.76%) 1808 (32.03%) 1061 (18.80%) 509 (9.02%)	135 (2.39%)
HRI (Heart Rate)	Numeric	(1) Less than 60 (bradycardia) (2) 120–129 (normal) (3) 130–139 (tachycardia)	167 (2.96%) 4521 (80.10%) 834 (14.78%)	122 (2.16%)
TEMPI (Temperature)	Numeric	(1) Less than 37.5 °C (2) 37.5–37.9 °C (3) 38–38.4 °C (4) Greater than 38.4 °C	4925 (87.26%) 471 (8.35%) 137 (2.43%) 74 (1.31%)	37 (0.66%)
Symptoms				
FEVER	2 categories	(1) Yes (2) No	1339 (23.72%) 4301 (76.20%)	4 (0.07%)
COUGH	2 categories	(1) Yes (2) No	2354 (41.71%) 3286 (58.22%)	4 (0.07%)
SPUTUM	2 categories	(1) Yes (2) No	1629 (28.86%) 4011 (71.07%)	4 (0.07%)
ST (Sore Throat)	2 categories	(1) Yes (2) No	879 (15.57%) 4761 (84.36%)	4 (0.07%)
RNR (Runny Nose)	2 categories	(1) Yes (2) No	618 (10.95%) 5022 (88.98%)	4 (0.07%)
MAM (Muscle Aches)	2 categories	(1) Yes (2) No	923 (16.35%) 4717 (83.58%)	4 (0.07%)
FM (Fatigue)	2 categories	(1) Yes (2) No	242 (4.29%) 5398 (95.64%)	4 (0.07%)
SOB (Shortness of Breath)	2 categories	(1) Yes (2) No	700 (12.40%) 4940 (87.53%)	4 (0.07%)
HEADA (Headache)	2 categories	(1) Yes (2) No	964 (17.08%) 4676 (82.85%)	4 (0.07%)

Table 1. *Cont.*

Clinical Variables	Type	Components	Patient (n = 5644)	Missing Values
Symptoms				
ACC *(Altered Consciousness)*	2 categories	(1) Yes (2) No	33 (0.58%) 5607 (99.34%)	4 (0.07%)
VN *(Vomiting)*	2 categories	(1) Yes (2) No	245 (4.34%) 5395 (95.59%)	4 (0.07%)
DIARR *(Diarrhea)*	2 categories	(1) Yes (2) No	521 (9.23%) 5119 (90.70%)	4 (0.07%)
Underlying Diseases				
DM *(Diabetes Mellitus)*	2 categories	(1) Yes (2) No	701 (12.42%) 4940 (87.53%)	3 (0.05%)
HTN *(Hypertension)*	2 categories	(1) Yes (2) No	1219 (21.60%) 4422 (78.35%)	3 (0.05%)
HF *(Heart Failure)*	2 categories	(1) Yes (2) No	59 (1.05%) 5581 (98.88%)	4 (0.07%)
CCD *(Chronic Cardiac Disease)*	2 categories	(1) Yes (2) No	184 (3.26%) 5441 (96.40%)	19 (0.34%)
ASTHMA	2 categories	(1) Yes (2) No	130 (2.30%) 5511 (97.64%)	3 (0.05%)
COPD *(Chronic Obstructive Pulmonary Disease)*	2 categories	(1) Yes (2) No	41 (0.73%) 5600 (99.22%)	3 (0.05%)
CKD *(Chronic Kidney Disease)*	2 categories	(1) Yes (2) No	56 (0.99%) 5585 (98.85%)	3 (0.05%)
MALIG *(Malignant Cancer)*	2 categories	(1) Yes (2) No	146 (2.59%) 5494 (97.34%)	4 (0.07%)
CLD *(Chronic Liver Disease)*	2 categories	(1) Yes (2) No	82 (1.45%) 5236 (92.77%)	326 (5.78%)
RDAD *(Rheumatism or Autoimmune Disease)*	2 categories	(1) Yes (2) No	38 (0.67%) 5274 (94.12%)	332 (5.88%)
DEMEN *(Dementia)*	2 categories	(1) Yes (2) No	227 (4.02%) 5088 (90.15%)	329 (5.83%)
Blood Test Results				
HGB *(Hemoglobin)*	Numeric	(1) Less than 11 (anemia) (2) 11–16 (normal) (3) Greater than 16 (elevated)	1504 (18.67%) 2970 (52.62%) 101 (1.79%)	1519 (26.91%)
HCT *(Hematocrit)*	Numeric	(1) Less than 41 (anemia) (2) 41–45 (normal) (3) Greater than 45 (elevated)	1317 (24.29%) 2553 (45.23%) 196 (3.47%)	1524 (27.00%)
LYMPHO *(Lymphocytes)*	Numeric	(1) Less than 20 (lymphocytopenia) (2) 20–40 (normal) (3) Greater than 40 (lymphocytosis)	934 (16.55%) 2487 (44.06%) 681 (12.07%)	1542 (27.32%)
PLT *(Platelets)*	Numeric	(1) Less than 150,000 (thrombocytopenia) (2) 150,000–450,000 (normal) (3) Greater than 450,000 (thrombocytosis)	524 (9.28%) 3530 (62.54%) 73 (1.29%)	1517 (26.88%)
WBC *(White Blood Cells)*	Numeric	(1) Less than 4000 (leukocytopenia) (2) 4000–11,000 (normal) (3) Greater than 11,000 (leukocytosis)	706 (12.51%) 3244 (57.48%) 177 (3.14%)	1517 (26.88%)

To address the missing values in our dataset, we implemented a missing value imputation algorithm based on the dataset statistics. For variables with a proportion of missing values of less than 10%, we imputed the most frequent value for categorical variables and the median for numerical variables. When the proportion of missing values exceeded 10%, particularly for the *PREG* and *PREGW* variables, we first classified these variables using *SEX*. Except for the instances in which the *PREGW* variable was already recorded, we assigned the remaining

values as *No* and *0 weeks* for *PREG* and *PREGW*, respectively. We used the KNN imputation method for the remaining missing values that exceeded the proportion of 10%. In KNN, the value of k, which represents the number of neighbors, significantly affects the imputation results. A small k value can make the model sensitive to noise, whereas a large k value can smooth the decision boundary, leading to the loss of important patterns in the data. To select the optimal k value, we used a scoring metric based on Jensen–Shannon divergence. This metric evaluates the quality of the imputed values by comparing them with the original distribution. Consequently, we selected $k = 2$, which yielded the smallest divergence. Figure 2 shows the imputed distribution of the KNN for the blood results, including *HCT*, *LYMPHO*, *PLT*, and *WBC*. Algorithm 1 describes the procedure for imputing missing values.

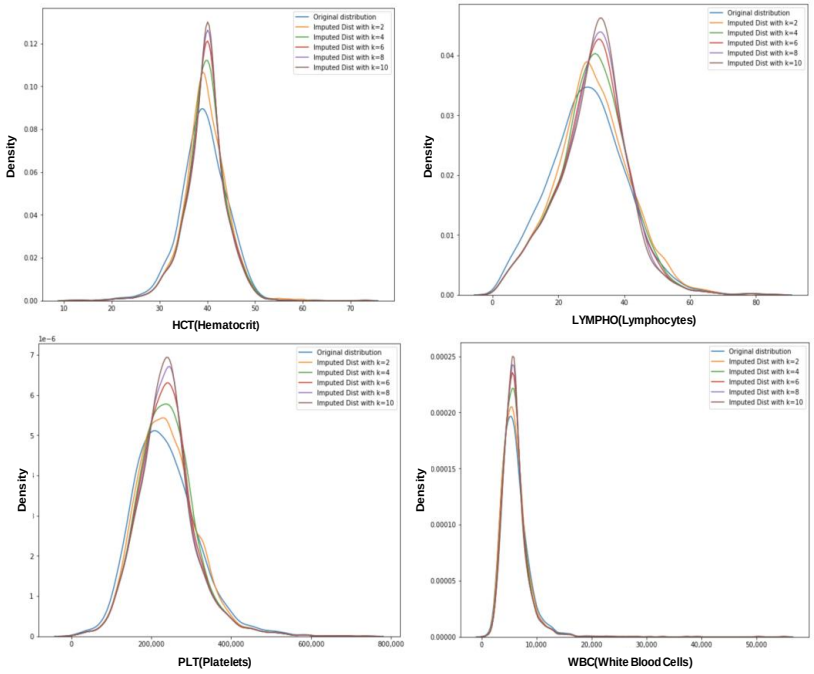


Figure 2. Imputed distribution of KNN for blood test results.

Algorithm 1. Missing Value Imputation

Input: x is data instances with missing data in the dataset; n is the size of the neighbor set
Output: updated x with imputed missing data

for *each missing value in x* **do:**
  if *missing values < 10%*
    if *x is categorical type data*:
      impute the missing value with the most frequent value
    elseif *x is numeric type data*:
      impute the missing value with the median value
  else if *missing values > 10%*:
    if *x is related to PREG*:
      if *SEX is male*:
        impute *PREG* as 'No' and *PREGW* as '0 weeks'
      else if *SEX is female*:
        if *PREGW records are empty*:
          impute *PREG* as 'No' and *PREGW* as '0 weeks'
        if *PREG is recorded as 'No'*:
          impute *PREGW* as '0 weeks'
end for
initialize *score* **as an** empty list
for *each number of neighbor set n* **do:**
  impute the remaining values using KNN approach
  score ← calculate divergence score for comparison with original distribution
end for
k ← index corresponding to *min(score)*
return imputed missing values using the selected k value

The scoring metric based on the Jensen–Shannon divergence, which is used for comparison with the original distribution, can be calculated as follows:

$$\begin{aligned} Score &= \sqrt{\tfrac{1}{2}(KL(p,\ m) + KL(q,m)}\ \ where \\ KL(p,\ m) &= \sum_{x\ \epsilon\ \Omega} p(x)\log\left(\tfrac{p(x)}{q(x)}\right),\ m = 0.5 * (p+q) \end{aligned} \tag{1}$$

where p denotes the original data distribution and q denotes the imputed data distribution from the KNN approach; m denotes the mixed distribution of p and q. This score measures the average divergence between the two distributions relative to the mixed distribution. A score of 0 indicates identical distributions, whereas a score of 1 indicates maximally different distributions.

We then performed standardization and mean encoding of the datasets. Standardization involves rescaling the distribution of values such that the mean of the observed values is 0 and the standard deviation is one. Thus, we ensured that all features, except *CSS*, followed a Gaussian normal distribution with a mean of 0 and standard deviation of 1. This procedure is calculated as follows:

$$Data_{stand} = \frac{Data - \mu(Data)}{\sigma(Data)} \tag{2}$$

where μ(train) and σ(train) represent the mean and standard deviation for each feature in the dataset, respectively. Following standardization, we performed a mean encoding of the categorical values. This procedure transforms the categorical values into the mean of the target variables for each category.

3.3. Data-Driven Strategy

Acquiring all clinical variables of a patient in real-world clinical settings can be challenging due to limitations related to the medical staff and facilities. To address these challenges, in this phase, we prioritize and group the clinical variables based on their ease of collection during patient visits to a hospital. We organized these variables into five categories: patient demographics, vital signs, symptoms, underlying diseases, and blood test results. Figure 3 illustrates our data-driven strategy.

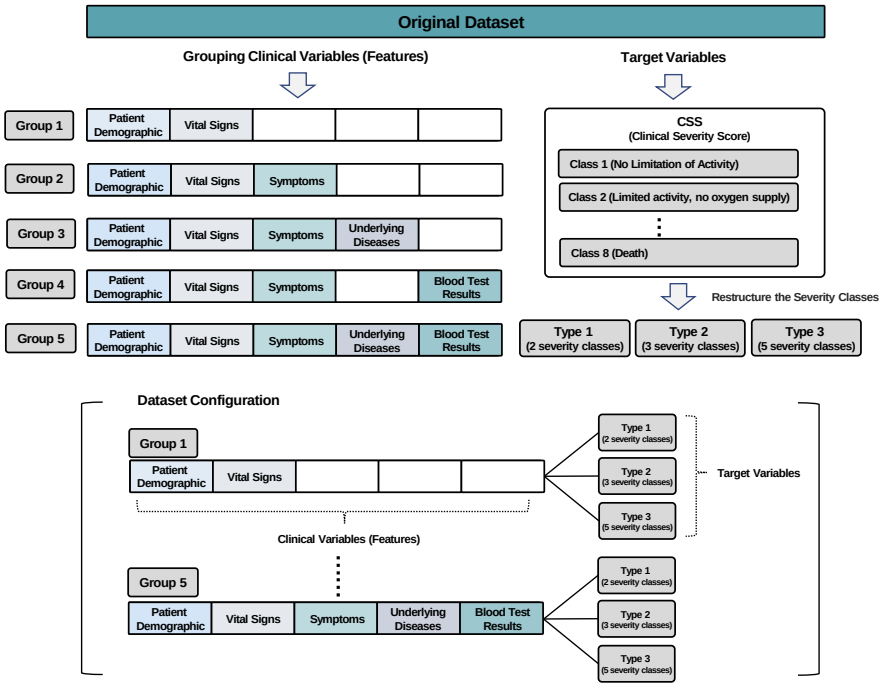


Figure 3. Description of data-driven strategy.

- Group 1 comprised nine clinical variables related to patient demographics and vital signs, including *AGE*, *BMI*, *SBP*, and *HRI*. These variables can be obtained simply by querying the patient and can be easily collected upon arrival at the hospital.
- Group 2 was obtained by expanding Group 1 by adding symptoms such as *FEVER* and *HEADA*, resulting in 21 clinical variables. These variables were binary data types that indicated whether a COVID-19 patient exhibited the related symptom.
- Group 3 was obtained by expanding Group 2 by adding 32 clinical variables, including underlying diseases related to the patient's current or past medical history, such as *DM* and *HTN*.
- Group 4 comprised 26 clinical variables and was obtained by adding blood test results, such as *HGB* and *LYMPHO*, to the Group 2 data in anticipation of difficulties accessing the patient's current/past medical history.
- Finally, Group 5 was based on Group 3 and included underlying diseases related to the patient's current/past medical history, along with the addition of blood test results, resulting in 37 clinical variables.

The target variable in our original dataset, *CSS*, was divided into eight different severity classes, as shown in Table 2. However, the original dataset contained a significant data imbalance. Out of 5664 patients, 4456 (78.85%) were classified as Class 1, leaving only 1188 patients (21.05%) across Classes 2–8. Furthermore, Classes 4–7 contained less than 1% of the total patient data. This data imbalance presented a challenging issue for model training because some classes were not adequately represented.

Table 2. Description of clinical severity score (CSS).

Severity Class	Original CSS (n = 5644)	Type 1 (2 Classes)	Type 2 (3 Classes)	Type 3 (5 Classes)
1	No limitation of activity (4456, 78.95%)	Low Severity (5312, 94.12%)	Mild (4791, 84.89%)	Asymptomatic (4456, 78.95%)
2	Limited activity but no oxygen supply required (335, 5.94%)			Mild (335, 5.94%)
3	Oxygen supply with nasal prong required (478, 8.47%)		Severe (599, 10.61%)	Moderate (521, 9.23%)
4	Oxygen supply with facial mask required (43, 0.76%)			
5	Non-invasive mechanical ventilation (36, 0.64%)	High Severity (332, 5.88%)		Severe (91, 1.61%)
6	Invasive mechanical ventilation (42, 0.74%)			
7	Multi-organ failure or ECMO (13, 0.23%)		Critical (254, 4.50%)	
8	Death (241, 4.27%)			Fatal (241, 4.27%)

To address these issues, we restructured the severity classes into three types, considering the various COVID-19 severity levels. These three types of restructured severity classes were utilized in model training as the target variables.

- Type 1 comprises two classes aimed at distinguishing between general ward and Intensive Care Unit (ICU) patients for efficient ward allocation. Patients from classes 1–4 in the original dataset were classified as *low-severity* (5312 patients, 94.12%) and patients from classes 5–8 were classified as *high-severity* (332 patients, 5.88%).
- For Type 2, the target variables were restructured into three classes. Classes 1 and 2 were considered *mild* (4791 patients, 84.89%), classes 3–6 were considered *severe* (599 patients, 10.61%), and classes 7 and 8 were considered *critical* (254 patients, 4.50%).
- Type 3 provided a more detailed classification by subdividing *low-* and *high-severity* levels into five classes. Class 1 was *asymptomatic* (4456 patients, 78.95%), Class 2 was *mild* (335 patients, 5.94%), Classes 3–4 were *moderate* (521 patients, 9.23%), Classes 5–7 were *severe* (91 patients, 1.61%), and Class 8 was associated with *fatal* (241 patients, 4.27%).

3.4. Feature Selection

In this phase, we propose a feature selection algorithm to improve the performance of our severity prediction model by using only selected features of high importance and removing variables of low relevance. Feature selection is an important process aimed at identifying a subset of relevant features to improve the ability to predict clinical severity and enhance interpretability. Feature selection methods can be classified into three main types: filters, wrappers, and embedded methods [45]. Filter methods use statistical techniques to evaluate the correlation or dependence between each input feature and the target outcome. These methods are efficient and fast; however, they can select redundant variables because they do not consider the interactions between the features. Moreover, a high correlation coefficient does not always indicate model suitability. In contrast, embedded methods select features based on the feature importance results of learning-based models, such as the Gradient Boosting Machine (GBM) or random forest (RF). Although these methods consider feature interactions, they may be overly biased toward specific models.

To address these limitations, we propose a feature selection algorithm that uses wrapper methods to select a subset of features based on various models with high accuracies. To identify the important features, we employed greedy search techniques, including forward selection, backward elimination, and RFECV. We used all three feature selection methods to determine the optimal feature set. The drawback of wrapper methods is that the selected features are heavily dependent on the specific model used for the feature selection. This implies that a feature set that is optimized for one model may not perform effectively with another model. To address the model dependency issue inherent in wrapper methods, our algorithm utilizes various tree-based ensemble models, such as Light Gradient Boosting Machine (LGBM), XGBoost, and CatBoost. Instead of relying on a single model, the proposed algorithm evaluates the performance of each feature across these models. It then compiles a feature set containing features that have high importance across multiple models, thereby reducing the reliance on feature selection for any single model. We also employed cross-validation methods to train these models iteratively and eliminated features deemed to have low importance. Algorithm 2 briefly describes the feature selection method.

Algorithm 2. Feature Selection

Input: x is the standardized and mean encoded datasets; y is the target variable on datasets
Output: *sf* is the selected features' sets

[**Evaluation Function**]: Internal function to evaluate the performance of a given set of features for each model
define *model dictionary M* (*M = {LGBM, XGBoost, and CatBoost}*)
function *evaluate_features* (*fs*, *x*, *y*)**:**
  initialize best score *bs* **as** 0.0
  for *each model in M* **do:**
   score ← mean cross-validation score for model using feature sets *fs*, *x* and *y*
   if *score* > *bs***:**
    bs ← *score*
  end for
  return *bs*
end function
[**Forward Selection**]: Iterate until no more features can be selected from full set of features
initialize forward-selected feature set *fsets*, and best global score *bgs* **as** 0.0
set remaining feature set *flists* **as** full set of feature lists
while *flists* is not empty**:**
 initialize *bestscore* **as** 0.0
 for each feature *f* in *flists* **do:**
   current feature set *cflist* ← *fset* + feature *f*
   score ← **call function** *evaluate_features (cflist, x, y)*
   if *score* > *bestscore***:**

Algorithm 2. *Cont.*

19: *bestscore* ←*score*, and *bf* ← feature *f*
20: **end for**
21: **if** *bestscore* > *bgs***:**
22: *bgs* ← *bestscore*, and *fsets* ← *fsets* + feature *f*
23: **remove** *bf* from *flights*
24: **else**
25: **break**
26: **end while**
[**Backward Elimination**]: Iterate until no more features can be removed from full set of features
27: **set** backward selected feature set *bsets* **as** full set of features
28: **initialize** best global score *bgs* **as** 0.0
29: **while** size of *bsets* > 1**:**
30: **for** each feature *f* in *bsets* **do:**
31: current feature set *cflist* ← [*f* **for** *f* **in** *bsets* **if** *f* not equal to feature *f*]
32: *score* ← **call function** *evaluate_features (cflist, x, y)*
33: **if** *score* > *bestscore***:**
34: *bestscore* ←*score*, and *bf* ← feature *f*
35: **end for**
36: **if** *bestscore* > *bgs*
37: *bgs* ← *bestscore*, and *bsets* ← *bsets* - feature *f*
38: **else**
39: **break**
40: **end while**
[**RFECV Selection**]: Features are recursively removed to find an optimal number that maximizes model performance.
41: **initialize** RFECV selected feature set *rfecvsets* **as an** empty list, best global score *bgs* **as** 0.0
42: **for** *each model in M* **do:**
43: apply RFECV to the model
44: *score* ← obtaining *maxscore* using *cvresults* of RFECV
45: if *score* > *bgs***:**
46: *bgs* ← *score*, and *rfecvsets* ← best features from RFECV
[**Merge and find feature lists**]: Merge the results of forward, backward, and RFECV and find common feature elements
47: **merge** *fsets*, *bsets*, and *rfecvsets* into *merged_list_L*
48: **count** feature occurrences in *merged_list_L*
49: *sf* ← features occurring at least twice in *merged_list_L*
50: **return** *sf*

Our feature selection algorithm provides a comprehensive approach to feature selection. First, we initialized machine learning models, such as LGBM, XGBoost, and CatBoost. The internal function *evaluate_features* evaluates the performance of the given features with five-fold stratified cross-validation and returns the best model and its corresponding score. We then used forward selection to select the best features and backward elimination to remove unnecessary features. Finally, we used RFECV to determine the optimal number of features for each model. After combining the features extracted using these three methods, we ultimately returned only the important features that appeared more than twice. Figure 4 shows the results of the selected feature importance based on the data-driven strategy, with blue bars indicating individual scores and red lines connecting these scores across the groups for comparison.

In Group 1, we selected five of nine clinical variables as important features. The feature importance values for severity prediction were ranked in the following order: *TEMPI*, *BMI*, *AGE*, and *HRI*. In Group 2, we identified 13 of the 21 clinical variables as significant. Among the symptoms, we can observe that *SOB* had the highest feature importance. This was followed by *AGE*, which was also emphasized in Group 1. Most of the symptoms in Group 2 had similar feature importance values. However, *FEVER* was slightly more significant than the other variables. For Group 3, we determined that 22 of 32 clinical variables were

important features. Clinical variables related to patient demographics and vital signs, such as *AGE* and *HRI*, were more important than those associated with underlying diseases. In Group 4, 16 of the 26 variables were selected as important features. Clinical variables related to blood test results, especially *WBC* and *LYMPHO*, showed high feature importance and significantly affected the severity prediction. Finally, in Group 5, we selected 30 of the 37 clinical variables as important features. The most critical clinical variables were related to blood test results, including *WBC*, *PLT*, *LYMPHO*, *HCT*, and *HGB*. Concurrently, patient demographics and vital signs, such as *AGE* and *TEMPI*, remained highly relevant.

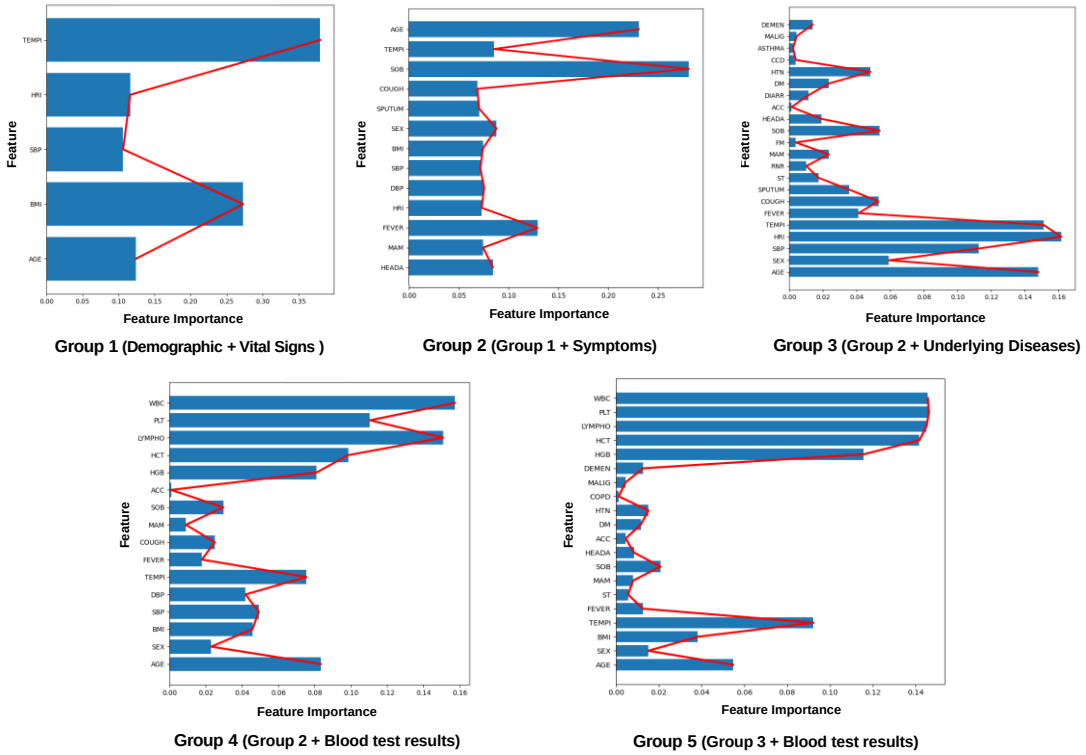


Figure 4. Results of selected feature importance according to the data-driven strategy.

However, certain clinical variables, such as *PREG*, *PREGW*, *VN*, *CLD*, and *RDAD*, were not selected for Groups 1–5. Consequently, they were identified as noncontributors to the severity prediction capabilities of the model.

3.5. Data Splitting and Oversampling

In this phase, we divide the dataset into training and testing subsets to develop a severity prediction model utilizing a subset of the selected features. We also address the issue of data imbalance by applying oversampling techniques to enhance the effectiveness of model training. Initially, we divided the patient dataset into training (3956/5651, 70%) and testing (1696/5, 651, 30%) datasets. To prevent bias during the model training, we randomly shuffled the training data and employed a stratified sampling technique. We also conducted five-fold stratified cross-validation, dividing the dataset into five folds. One fold was used as the validation dataset, and the remaining four as the training dataset. To ensure comprehensive validation, we repeated this cross-validation process 10 times. Each fold served multiple times as both the training and validation dataset, significantly improving the model's robustness and reliability by extensively evaluating it against various data subsets.

As described in Table 2, our target variable, CSS, has a severe data imbalance. Such an imbalance can be problematic for machine-learning algorithms, especially those based on optimization techniques, as they can be biased toward classes with a greater frequency of occurrence. This can negatively affect the predictive performance of minority classes.

To mitigate this imbalance, we utilized two oversampling techniques commonly used in medical research: the Synthetic Minority Oversampling Technique (SMOTE) and Adaptive Synthetic Sampling (ADASYN) [46–48]. SMOTE synthesizes the data points for a minority class by utilizing the Euclidean distance to the nearest neighbors. Based on the original attributes, these new samples are very similar to the actual data. However, ADASYN generates synthetic samples by considering the distance between the majority and minority classes, thereby enhancing the minority class data. This method is often considered more sophisticated than SMOTE.

It is important to only perform oversampling on the training data and not on the validation or test sets. Oversampling before splitting the dataset can lead to duplication or the synthesis of instances in the validation or test sets, which can inflate the performance metrics. Therefore, we applied oversampling only to the training dataset, which contained the data for 3956 patients. Table 3 lists the oversampling ratios for Types 1, 2, and 3 of the target variable *CSS* using SMOTE and ADASYN.

Table 3. Oversampling ratio of Clinical Severity Score (CSS).

Severity Class		Before Oversampling (n = 3956)	After Oversampling (n = 5585)
Type 1	Low-Severity	3723 (94.12%)	3723 (66.66%)
	High-Severity	233 (5.88%)	1862 (33.34%)
Type 2	Mild	3358 (84.89%)	3358 (60.13%)
	Severe	420 (10.61%)	1500 (25.87%)
	Critical	178 (4.50%)	782 (14.00%)
Type 3	Asymptomatic	3123 (78.95%)	3123 (55.92%)
	Mild	235 (5.94%)	635 (11.37%)
	Moderate	365 (9.23%)	765 (13.70%)
	Severe	64 (1.61%)	493 (8.83%)
	Fatal	169 (4.27%)	569 (10.19%)

3.6. Stacking Ensemble Model Construction

In this phase, we propose an adaptive stacking ensemble technique that automatically combines multiple single classifiers. Ensemble techniques generate multiple weak classifiers from a given training dataset, and then combine them to form a single strong classifier. This approach can reduce the errors owing to bias and variance that can occur when using a single model. As a result, ensemble techniques provide a better predictive performance than that achieved using individual models. Ensemble techniques are broadly classified into bagging, boosting, and stacking techniques, as illustrated in Figure 5.

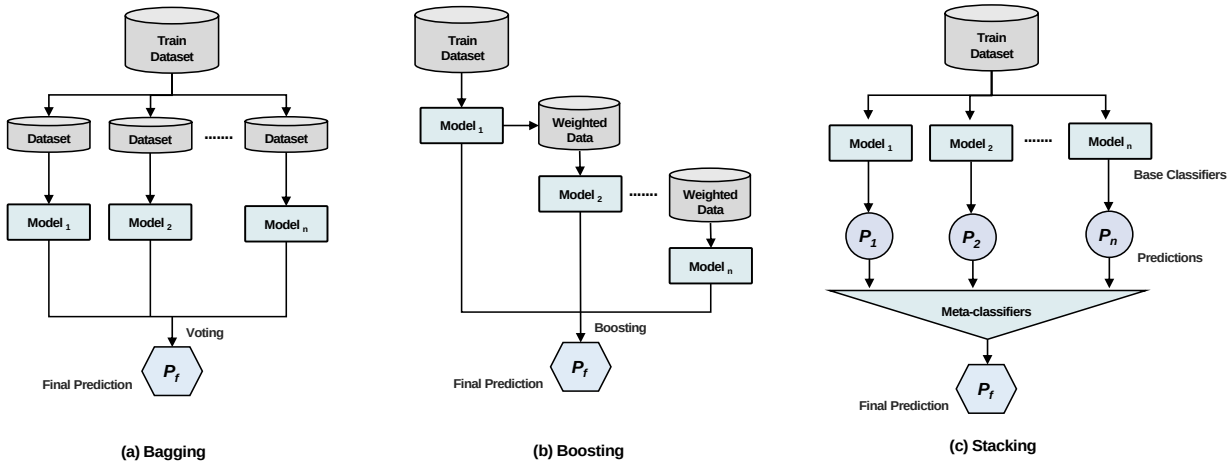


Figure 5. Ensemble techniques for bagging, boosting, and stacking.

During bagging, the training data are randomly resampled to form multiple subsets of the same size. Weak classifiers trained on these subsets are combined using a voting mechanism for the final prediction. Bagging reduces the variance to improve the model performance but cannot effectively address model bias. Additionally, each model has the same weight in the final vote, which can limit the flexibility in certain applications.

In contrast, the boosting technique considers the multiple generated models to be unequal and reflects the weights assigned to each model in the final prediction vote. Initially, the models were trained using all available training data. Subsequently, the predictive performance was evaluated to adjust the weights of the training samples. The well-classified data points receive lower weights, whereas the poorly classified data points receive higher weights. The models are then trained sequentially on the weighted data samples. The final prediction is made using a weighted vote that includes models with varying confidence levels. Boosting improves the model performance by adjusting the bias; however, it has drawbacks, such as a slower learning speed and vulnerability to overfitting, especially in the presence of noisy data.

The stacking technique generates a meta-classifier by learning from the predictions of two or more base classifiers. This technique typically performs better than a single model because it can leverage the individual strengths and weaknesses of each base classifier. However, stacking is computationally intensive and can lead to overfitting if not carefully tuned. Despite these difficulties, stacking can provide a superior predictive performance when correctly executed. This approach is particularly useful in handling the outliers that can negatively affect a single model.

In this paper, we employed a multilayer stacking ensemble approach that integrates different base classifiers over multiple layers under the assumption that the learning outcomes of the individual models are independent. Figure 6 shows the model construction process for the proposed adaptive stacking ensemble technique.

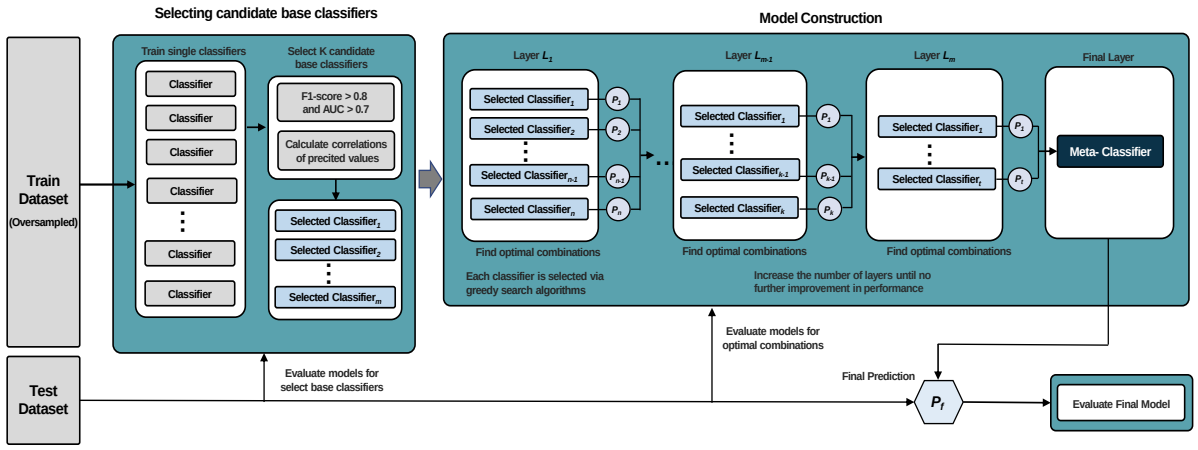


Figure 6. Model construction process of the proposed adaptive stacking ensemble technique.

Initially, the module for '*selecting candidate base classifiers*' identifies the classifiers that should be included in the stacking ensemble. At this stage, m single classifiers are trained. The trained classifiers are then evaluated using five-fold stratified cross-validation to measure their average F1 and AUC scores. Based on a predefined threshold, only k candidate classifiers with an F1 score or AUC score of 0.7 or above are selected for inclusion. Subsequently, the correlations between the predicted outcomes of the selected classifiers are calculated. To reduce model bias and create a diverse set of predictions, classifiers with low prediction correlations are chosen. Classifier combinations with a correlation coefficient exceeding 0.8 are excluded, and models with poor performance

for certain pairs are removed. The equation used to calculate the correlation coefficient between two classifiers is defined as follows:

$$M_{corr}(M_x, M_y) = \frac{\sum_{i=1}^{n} \left(M_x - \overline{M_x}\right)\left(M_y - \overline{M_y}\right)}{\sqrt{\sum_{i=1}^{n} \left(M_x - \overline{M_x}\right)^2 \left(M_y - \overline{M_y}\right)^2}} \quad (3)$$

where M_x and M_y represent the prediction outcomes from the individual classifiers x and y, respectively, and $\overline{M_x}$ and $\overline{M_y}$ represent the mean values of the predicted outcomes of these classifiers. Algorithm 3 describes the procedure for selecting the candidate base classifiers.

Algorithm 3. Selection of candidate base classifiers

Input: x is the oversampled train dataset, y is the test dataset
Output: N selected classifiers for composite stacking ensemble

define *model dictionary M* (set of *m single classifiers*)
initialize model's evaluation score F_m **as a** dictionary
initialize selected classifiers N **as an** empty list
for *each model in M* **do:**
  tm ← train each model using dataset x
  F1_score, AUC_score ← evaluate *tm* with dataset y using 5-fold stratified cross-validation
  F_m[*model*] ← (*F1_score, AUC_score)*
end for
for *each model, (F1_score, AUC_score) in* F_m **do:**
  if either *F1 score* or *AUC_score* ≥ threshold:
    add model to select classifiers N
end for
for *each model* M_i *in* N **do:**
  for *each model* M_j *in* N **do:**
    if *model* M_i != *model* M_j
      $M_{corr}\left(M_i, M_j\right)$ ← calculate correlation coefficient between M_i and M_j
      if $M_{corr}\left(M_i, M_j\right)$ ≥ corr_threshold**:**
        if F_m[*model* M_i] > F_m[*model* M_j]**:**
          mark M_j for removal
        else:
          mark M_i for removal
  end for
end for
remove marked models from selected classifiers N
return N

Our adaptive stacking ensemble model combines the individual base classifiers to learn the best possible prediction combination of various base classifiers and is structured as a multilayer stacking ensemble model that outperforms the individual models. In the '*model construction module*', the N candidate base classifiers selected from the previous module are first sorted in descending order based on their F1 and AUC scores. The training and testing datasets are split to facilitate the construction of a multilayered structure. A greedy algorithm utilizing an internal function called the *train_layer* is used to progressively add the classifiers based on their performance and evaluate the efficiency of each combination. If adding more classifiers does not enhance the performance, the current combination is considered the optimal single-layer configuration and the process moves to the next layer. Thus, the classifier combinations in the previous layer serve as inputs to the next layer. This iterative process of adding classifiers and layers is continued until no further performance improvement is observed. Due to the expected high complexity of the multilayer stack ensemble model, Logistic Regression was chosen as the meta-learner to minimize the risk of data overfitting. Unlike traditional methods, where the user manually selects the classifiers that will be integrated into the stacking ensemble, our approach automatically

selects the base classifiers using a greedy algorithm. Algorithm 4 describes the procedure for constructing the adaptive stacking ensemble model.

Algorithm 4. Adaptive stacking ensemble model construction

Input: N selected classifiers for composite stacking ensemble, x is the oversampled train dataset, y is the test dataset
Output: Final F1_score *final_f1*

define *model dictionary N* (set of *selected candidate base classifiers*)
initialize sort models by F1 and AUC scores S_m
initialize last best F1 score *lbf1* **as** 0.0
initialize $X_{next\_layer\_train}$ **as** X_{train}
initialize $X_{next\_layer\_test}$ **as** X_{test}
$S_m \leftarrow$sort N selected candidate classifiers by *F1_score* and *AUC_score*
randomly split train dataset x and test dataset y into (X_{train}, Y_{test}), (X_{train}, Y_{test})
function *train_layer* (*Models*, X_{train}, Y_{train})**:**
  initialize *layer_output* as empty list
  for *each model in Models* **do:**
    tm $\leftarrow$ train each model using datasets X_{train} and Y_{train}
    add *tm* to *layer_output*
  end for
  return *layer_output*
end function
initialize *patience_counter* **as** 0
while *true***:**
  initialize current best models *cbm* **as an** empty list
  initialize current best f1 *cbf1* **as** *lbf1*
  initialize interim models *im* **as an** empty list
  for *each model in* S_m **do**:
    add model to *im*
    *interim_* X_{train} $\leftarrow$ **call function** *train_layer (im,* $X_{next\_layer\_train}$, Y_{train})
    initialize *meta_model* as *LogisticRegression*
    train *meta_model* using *interim_* X_{train} and Y_{train}
    interim_f1 $\leftarrow$evaluate *meta_model* using $X_{next\_layer\_test}$ and Y_{test}
    if *interim_f1* > *cbf1***:**
      cbf1 $\leftarrow$ *interim_f1*
      cbm $\leftarrow$ copy interim models *im*
  end for
  if *cbf1* $\leq$ *lbf1***:**
    patience_counter += 1
    if *patience_counter* $\geq$ *max_pateince*:
      break
  else:
    lbf1 = *cbf1*
    $X_{next\_layer\_train}$ $\leftarrow$ **call function** *train_layer (cbm,* $X_{next\_layer\_train}$, Y_{train})
    $X_{next\_layer\_test}$ $\leftarrow$ **call function** *train_layer (*$X_{next\_layer\_test}$, Y_{test})
    patience_counter **to** 0
end while
initialize *final_meta_model* as *LogisticRegression*
train *final_meta_model* with $X_{next\_layer\_train}$ and Y_{train}
final_f1 $\leftarrow$ evaluate *final meta_model* using $X_{next\_layer\_test}$ and Y_{test},
return *final_f1*

We constructed an adaptive stacking model using 14 candidate base classifiers, including linear, nonlinear, distance-based, probabilistic, tree-based, and ensemble techniques. To effectively calibrate the parameters and ensure the optimal selection of hyperparameters for each classifier, we employed GridSearchCV in our hyperparameter tuning process. We systematically explored a wide range of hyperparameter combinations for each classifier,

within the context of a five-fold cross-validation process repeated 10 times. Table 4 lists the 14 classifiers we utilized, along with their respective hyperparameter settings.

Table 4. Candidate base classifiers and hyperparameter settings.

Type	Classifier Name	Parameter Setting
Linear	Logistic Regression (LR)	C: 0.01, solver: newton-cg, Max_iter: 1000, Penalty: l2
	Support Vector Machine 1 (SVM 1)	Kernel: Linear, C: 0.025
	Stochastic Gradient Descent (SGD)	Penalty: l2, Max_iter: 1000
Non-linear	Support Vector Machine 2 (SVM2)	Kernel: RBF, Gamma: 2, C: 1
	Multi-Layer Perceptron (MLP)	N_layers: 5 (including input, 3 FC hidden, and output layers) Hidden_layer_size: (100), alpha: 1.0, Max_iter: 1000
Distance-based	K-Nearest Neighbor (KNN)	N_neighbors: 2, weight: distance
Probabilistic	Gaussian Naïve Bayes (GNB)	Var_smoothing: 1×10^{-9}
Tree-based	Extra Trees Classifier (ExTree)	Criterion: entropy, Max_depth: 20, Min_sample_split: 2
	Decision Tree (DT)	Criterion: entropy, Max_depth: 20, Min_sample_split: 2
Ensemble	XGBoost (XGB)	N_estimators: 340, Max_depth: 2, Learning_rate: 0.0628, Gamma: 1.17
	Light Gradient Boosting Machine (LGBM)	N_estimators: 440, Max_depth: 10, Learning_rate: 0.0496, Max_depth: 6
	AdaBoost (ADA)	N_estimators: 320, Learning_rate: 0.0726
	Random Forest (RF)	N_estimators: 600, Max_depth: 8, N_jobs: -1, Min_sample_split: 2
	CatBoost (CAT)	Learning_rate: 0.7462

As depicted in Figures 7–9, the adaptive stacking ensemble learning process selects a subset of suitable classifiers from 14 base classifiers. This selection is based on a data-driven strategy defined by the groups of clinical variables for severity classification and the types of severity classes (i.e., *CSS*). To ensure robust performance, the selected classifiers were structured in a multi-layered architecture for prediction.

Figure 7 shows the structure of the Type 1 prediction, which comprises Groups 1–5 and predicts only two severity classes using the data-driven strategy. For the optimal combination of base classifiers, 8 out of 14 base classifiers (i.e., LGBM, XGB, DT, CAT, LR, RF, GNB, and MLP) were selected. All candidate base classifier models were structured in a two-layer architecture; RF, CAT, and LGBM were frequently selected. Additionally, for Groups 3 and 5, which included underlying diseases, the first layer consisted of three models.

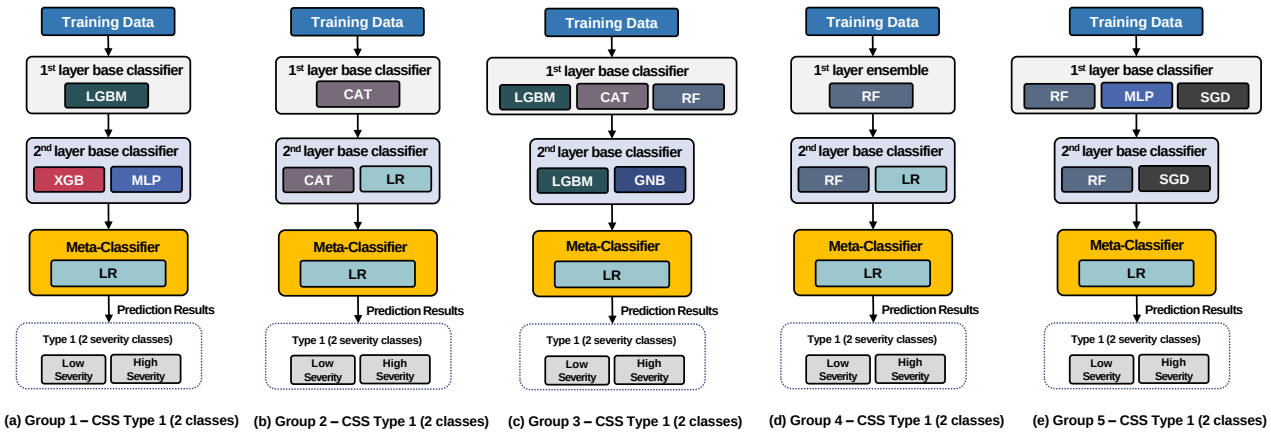


Figure 7. Optimal combinations of base classifiers in the adaptive stacking ensemble models for CSS Type 1.

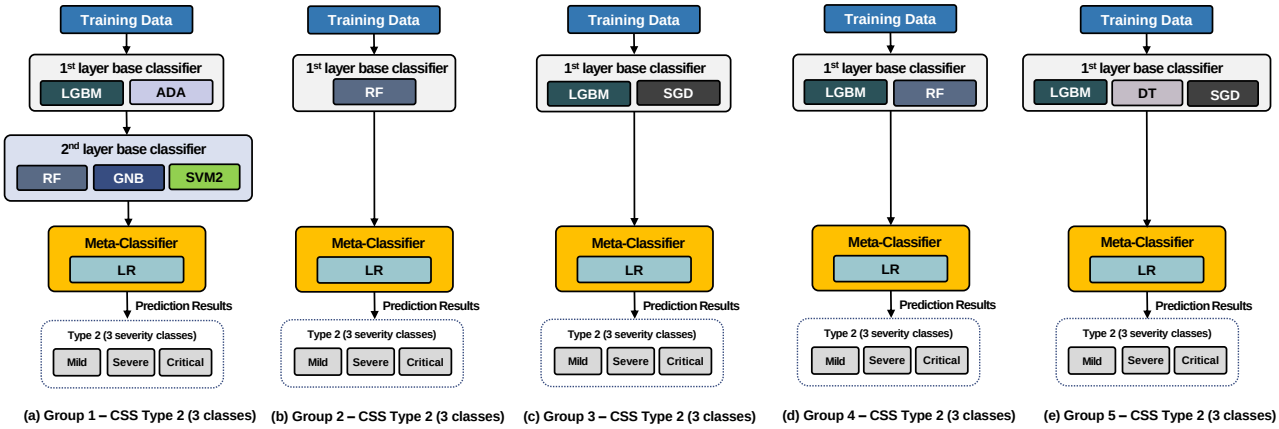


Figure 8. Optimal combinations of base classifiers in the adaptive stacking ensemble models for CSS Type 2.

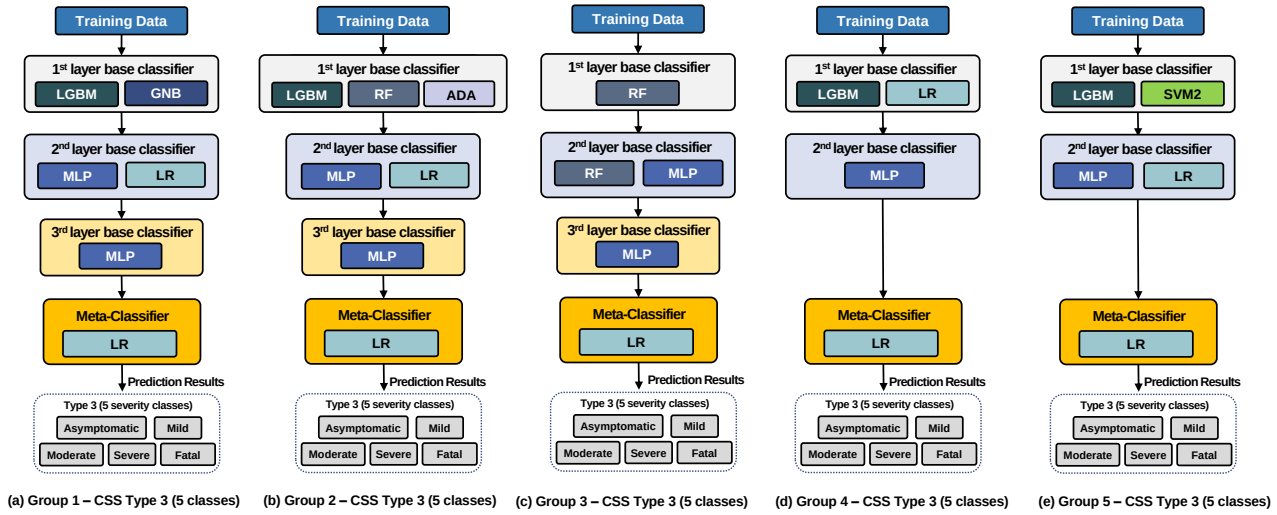


Figure 9. Optimal combinations of base classifiers in the adaptive stacking ensemble models for CSS Type 3.

Figure 8 shows the structure of Type 2 prediction, which encompasses Groups 1–5 and predicts three severity classes. This type expands the prediction scope when compared with that of Type 1, with three severity classes instead of two. For the optimal com-bination of base classifiers, 7 out of 14 classifiers (i.e., LGBM, ADA, RF, GNB, SVM2, SGD, and DT) were selected for Type 2. Except for the Group 1 model, which had fewer features (i.e., 9 clinical variables), the majority of the Type 2 models comprised the first layer. LGBM and RF were frequently selected as the base classifiers, whereas the remaining base classifiers were evenly incorporated into the combinations.

Figure 9 depicts the structure of Type 3 prediction, which is composed of Groups 1–5 and predicts five severity classes. Considering the complexity of predicting the five severity classes, the base classifiers were structured with three layers for Groups 1–3, whereas Groups 4 and 5, which included the blood test results, were structured with two layers. For the optimal combination of the base classifiers, 7 out of 14 classifiers (i.e., LGBM, GNB, LR, MLP, RF, ADA, and SVM2) were selected. MLP emerged as the most frequently selected base classifier across all group combinations for Type 3, and LGBM and LR were utilized in all groups except Group 3.

4. Results

In this section, we present a detailed description of the performance evaluation of our adaptive stacking ensemble technique. We first introduce the metrics used to evaluate our

technique and discuss the results of the performance evaluations. The performance of our technique was evaluated through three experiments: (1) the performance of the prediction models, (2) the effectiveness of the data-driven strategy and oversampling, and (3) the effect of the feature selection algorithm on the performance.

4.1. Evaluation Metrics

We utilized a standard set of evaluation metrics for classification problems, including precision, recall, F1 score, specificity, and AUC score. The precision, recall, specificity, and F1 score were derived from a confusion matrix that classified the predictions into four categories: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). Specifically, TP represents the number of correctly predicted positive instances, TN represents the number of correctly predicted negative instances, FP represents the number of negative instances incorrectly predicted as positive, and FN represents the number of positive instances incorrectly predicted as negative.

Precision is defined as the ratio of correctly predicted positive instances to all instances predicted as positive. It provides a measure of the accuracy of positive predictions. The equation used to determine precision is defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

Recall, also known as the True Positive Rate (TPR), quantifies the ratio of correctly predicted positive cases to all actual positives. It offers insights into the model's capacity to identify and retrieve relevant instances. The equation for recall is defined as follows:

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

The F1 score, which is the harmonic mean of the precision and recall, provides a balance between these two metrics and is particularly useful in scenarios where one metric is more valuable than the other. The equation for the F1 score is defined as follows:

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6}$$

In contrast, specificity represents the proportion of actual negative cases that are correctly classified. The equation for specificity is formally defined as follows.

$$Specificity = \frac{TN}{TN + FP} \tag{7}$$

The AUC score quantifies the entire two-dimensional area under the Receiver Operating Characteristic (ROC) curve, providing a comprehensive performance measure across all possible classification thresholds. The AUC score is formally defined as the integral of the ROC curve, which is the area between the ROC curve and False Positive Rate (FPR) axis. The equation is defined as follows:

$$\begin{gathered} AUC = \int_0^1 TPR\left(FPR^{-1}(x)\right)dx \ \ where \\ FPR = 1 - Specificity = \frac{FP}{TN+FP} \end{gathered} \tag{8}$$

where $TPR\left(FPR^{-1}(x)\right)$ represents the TPR as a function of the FPR. It maps the FPR to TPR to define the ROC curve. The integral spans from zero to one, effectively capturing the entire area under the ROC curve.

4.2. Performance of the Prediction Models

To evaluate our proposed technique, we conducted comparative experiments across three clinical severity classes using five groups of clinical variables, as defined by a data-driven strategy. Specifically, we carried out three sets of experiments: (1) Type 1 severity prediction for Groups 1–5; (2) Type 2 severity prediction for Groups 1–5; (3) Type 3 severity prediction for Groups 1– 5. The performance evaluation utilized a dataset comprising 1695 patients, which was divided during the data splitting and feature selection process. Moreover, we conducted comparative analyses with 14 well-known single classifiers, including Logistic Regression and XGBoost models. These models have previously been applied in studies utilizing the KDBC dataset to predict COVID-19 severity in South Korea [23,49,50]. To compare our results with those of more advanced methods, we also conducted experiments with two AutoML approaches: PyCaret, and H2O.ai [51,52]. PyCaret offers 16 classifiers, such as Linear Discriminant Analysis (LDA) and the Gradient Boosting Classifier (GBM), and streamlines the overall machine learning process by identifying optimal classifiers through efficient tuning. H2O.ai offers 18 classifiers, such as Distributed Random Forest (DRF) and the Generalized Linear Model (GLM), and enables the identification of the optimal classifiers or the automatic generation of stacking ensembles by combining models. For each model included in the performance evaluation, a total of 20 experiments were conducted, and the average performance results were recorded. Through these extensive evaluation processes, we comprehensively assessed each model's effectiveness in accurately predicting the various severity levels of COVID-19.

Table 5 presents the average performance results of our adaptive stacking ensemble model in comparison to other single classifiers and AutoML approaches for predicting Type 1 severity across all groups. The highlighted sections in each table indicate the highest-performing classifiers and their corresponding scores.

Table 5. Average performance results of Type 1 severity for all groups.

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Average	Logistic Regression (LR)	0.9426	0.8325	0.8832	**0.8302**	0.9203
	XGBoost (XGB)	0.9222	0.9048	0.9134	0.6700	0.8802
	PyCaret	0.9140	0.9064	0.9101	0.6785	0.8655
	H2O.ai	0.9354	0.8954	0.9148	0.8165	0.8423
	Average of All Single Classifiers and AutoML	0.9178	0.8737	0.8946	0.6776	0.8494
	Proposed Model (CSS Type 1)	**0.9582**	**0.9595**	**0.9588**	0.7634	**0.9380**

The prediction of Type 1 severity across Groups 1–5 employs a binary classification to distinguish between general and ICU patients. Our proposed model demonstrated considerable robustness, achieving an average F1 score of 0.9588 and AUC score of 0.9380 across all groups. Compared to existing single classifiers, Logistic Regression achieved an average F1 score of 0.8832 and an AUC of 0.9203, indicating our model's improvement of 7.56% in F1 and 1.77% in AUC scores. XGBoost achieved an F1 score of 0.9134 and an AUC of 0.8802, with our model showing an improvement of 4.54% in F1 and 5.78% in AUC scores. Moreover, our model significantly outperformed AutoML approaches. Compared to PyCaret, which recorded an F1 score of 0.9101 and an AUC of 0.8655, our model demonstrated improvements of 5.35% in F1 and 8.37% in AUC scores. Similarly, compared to H2O.ai, which achieved an F1 score of 0.9180 and an AUC of 0.8423, our model showed improvements of 4.40% in F1 and 9.57% in AUC scores. Furthermore, when compared to the average of all single classifiers and AutoML approaches, which had an F1 of 0.8946 and an AUC of 0.8494, our model showed an improvement of 7.18% in F1 and 10.43% in AUC scores for Type 1 severity. The enhanced performance of our model, especially evident in its recall and F1 score, signifies its effectiveness in accurately identifying patients at risk of severe COVID-19 outcomes.

For a more detailed understanding, Table 6 provides a summary of the comparative performance results for Type 1 severity across Groups 1–5. Comprehensive performance results of the comparisons with all 14 individual classifiers and AutoML approaches can be found in Table A1 of Appendix A.

Table 6. Summary of performance results of Type 1 severity for groups 1–5.

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Group 1	Logistic Regression (LR)	**0.9416**	0.7740	0.8496	**0.7604**	**0.8989**
	XGBoost (XGB)	0.8883	0.8735	0.8808	0.4644	0.8007
	Light Gradient Boosting Machine (LGBM)	0.8932	0.8770	0.8850	0.5059	0.8105
	PyCaret (CatBoost (CAT))	0.9188	0.8997	0.9091	0.6092	0.7849
	H2O.ai (Gradient Boosting Machine (GBM))	0.9347	0.9101	0.9223	0.7278	0.8406
	Proposed Model (Group 1-CSS Type 1) (Stacking: 1st layer: LGBM, 2nd layer: XGB, MLP, Meta: LR)	0.9262	**0.9301**	**0.9281**	0.6026	0.8722
Group 2	Logistic Regression (LR)	**0.9479**	0.7939	0.8641	**0.8168**	**0.9327**
	XGBoost (XGB)	0.9040	0.8876	0.8957	0.5823	0.8495
	CatBoost (CAT)	0.9084	0.8847	0.8974	0.5759	0.8645
	PyCaret (Random Forest (RF))	0.8993	0.8933	0.8963	0.5423	0.8462
	H2O.ai (Stacking: 1st layer: 1 GBM, 1 GLM, 2 DRF, Meta: GLM)	0.9354	0.8592	0.8957	0.8096	0.8410
	Proposed Model (Group 2-CSS Type 1) (Stacking: 1st layer: CAT, 2nd layer: CAT, LR, Meta: LR)	0.9407	**0.9421**	**0.9414**	0.7214	0.9098
Group 3	Logistic Regression (LR)	**0.9497**	0.8129	0.8760	**0.8180**	0.8880
	XGBoost (XGB)	0.9054	0.8916	0.8984	0.5619	0.8578
	AdaBoost (ADA)	0.9098	0.8972	0.9035	0.5554	0.8777
	PyCaret (CatBoost (CAT))	0.9032	0.9075	0.9053	0.5237	0.8532
	H2O.ai (Gradient Boosting Machine (GBM))	0.9358	0.8760	0.9049	0.8085	0.8426
	Proposed Model (Group 3-CSS Type 1) (Stacking: 1st layer: LGBM, CAT, RF, 2nd layer: LGBM, GNB, Meta: LR)	0.9471	**0.9474**	**0.9472**	0.6526	**0.9129**
Group 4	Logistic Regression (LR)	0.9363	0.8850	0.9099	0.8914	0.9412
	XGBoost (XGB)	0.9573	0.9363	0.9467	0.8741	0.9469
	CatBoost (CAT)	0.9575	0.9363	0.9468	0.8741	0.9479
	PyCaret (Logistic Regression (LR))	0.9313	0.8997	0.9152	0.8852	0.9410
	H2O.ai (Stacking: 1st layer: 1 GBM, 1 GLM, 2 DRF, Meta: GLM)	0.9357	0.9136	0.9245	0.8685	0.8439
	Proposed Model (Group 4-CSS Type 1) (Stacking: 1st layer: RF, 2nd layer: RF, LR, Meta: LR)	**0.9871**	**0.9872**	**0.9871**	**0.8959**	**0.9774**
Group 5	Logistic Regression (LR)	0.9376	0.8965	0.9166	0.8646	0.9407
	XGBoost (XGB)	0.9561	0.9350	0.9454	0.8671	0.9460
	Random Forest (RF)	0.9588	0.9389	0.9487	0.8329	0.9467
	PyCaret (CatBoost (CAT))	0.9176	0.9318	0.9246	0.8321	0.9024
	H2O.ai (Stacking: 1st layer: 1 GBM, 1 GLM, 2 DRF, Meta: GLM)	0.9355	0.9180	0.9267	0.8679	0.8433
	Proposed Model (Group 5-CSS Type 1) (Stacking: 1st layer: RF, MLP, SGD 2nd layer: RF, SGD, Meta: LR)	**0.9901**	**0.9908**	**0.9904**	**0.9443**	**0.9877**

Based on a detailed observation of each group's performance, the proposed model's severity prediction for Group 1 was lower than that of other groups, with an F1 score of 0.9281 and AUC score of 0.8772. This lower performance is due to Group 1 considering only nine clinical variables related to demographics and vital signs, with feature selection

further reducing this to five variables, thus offering fewer features compared to other groups. Consequently, the performance of other single classifiers and AutoML approaches in Group 1 was also observed to be lower. In Group 2, which was formed by expanding Group 1 through the addition of symptoms, the proposed model showed a performance improvement of 1.33% in the F1 score and 3.76% in the AUC score when compared with the results for Group 1. Other single classifiers and AutoML approaches also demonstrated that the inclusion of symptoms has a positive impact on prediction performance. In Group 3, the addition of symptoms and past or current medical history as clinical variables resulted in a performance improvement of 1.91% in the F1 score and 4.07% in the AUC score when compared with the results for Group 1. However, a comparison of Groups 3 and 2 shows that the inclusion of past or current medical history did not significantly enhance the severity prediction performance of Group 3. The improvement was only 0.58% in the F1 score and 0.31% in the AUC score. Other single classifiers and AutoML approaches also showed a less than 1% rise in the F1 score, and for logistic regression, a decrease in AUC scores was observed. This indicates that including a patient's past medical information in situations where patient symptom information is available does not significantly enhance the severity prediction performance. In Groups 4 and 5, which utilized blood test results, there was a significant improvement in the performance of all models when compared with the results of Groups 2 and 3. Specifically, the proposed model demonstrated exceptional performance in Groups 4 and 5, with F1 scores of 0.9871 and 0.9899 and AUC scores of 0.9974 and 0.9977, respectively. These results underscore the importance of blood test results for accurately predicting the severity of COVID-19. However, the AutoML approaches exhibited a relatively lower performance in Groups 4 and 5, showing only slight improvements or even decreases in AUC scores from Group 4 to Group 5. This observation confirms the feasibility and effectiveness of our proposed model for predicting COVID-19 severity.

Table 7 presents the average performance results of the proposed adaptive stacking ensemble model in comparison to other single classifiers and AutoML approaches for predicting Type 2 severity across all groups.

Table 7. Average performance results of Type 2 severity for all groups.

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Average	Logistic Regression (LR)	0.8239	0.7264	0.7758	**0.8014**	0.8148
	XGBoost (XGB)	0.8225	0.8283	0.8252	0.5668	0.8227
	PyCaret	0.8024	0.8324	0.8170	0.6889	0.7971
	H2O.ai	0.8112	0.8458	0.8281	0.6903	0.7775
	Average of All Single Classifiers and AutoML	0.8120	0.7612	0.7770	0.6200	0.7658
	Proposed Model (CSS Type 2)	**0.8717**	**0.8724**	**0.8720**	0.6930	**0.8198**

The prediction of Type 2 severity was classified into three classes: '*mild*', '*severe*', and '*critical*'. This detailed classification increased complexity compared to Type 1, resulting in an 8.6% decrease in the average F1 score and 10.5% decrease in the average AUC score for our proposed model. Despite these challenges, the proposed model demonstrated a superior performance in predicting Type 2 severity relative to other single classifiers and AutoML approaches. Despite these challenges, the proposed model demonstrated a superior performance in predicting Type 2 severity compared to other single classifiers and AutoML approaches. Specifically, the proposed model achieved an F1 score of 0.8720 and an AUC score of 0.8198, outperforming the averages of other models, including logistic regression, XGBoost, and two AutoML approaches. This underscores our model's robustness and effectiveness in more complex classification scenarios, highlighting its utility in accurately predicting the severity of COVID-19.

Table 8 provides a summary of the comparative performance results for Type 2 severity across Groups 1–5. Comprehensive performance results of the comparisons with all 14 individual classifiers and AutoML approaches can be found in Table A2 of Appendix A.

Table 8. Summary of performance results of Type 2 severity for Groups 1–5.

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Group 1	Logistic Regression (LR)	0.8060	0.7186	0.7598	**0.7670**	**0.7632**
	XGBoost (XGB)	0.7690	**0.8243**	0.7957	0.4468	0.7470
	Light Gradient Boosting Machine (LGBM)	0.7783	0.8162	0.7968	0.5067	0.7569
	PyCaret (AdaBoost (ADA))	0.7971	0.8226	0.8096	0.5954	0.7299
	H2O.ai (Generalized Linear Modeling (GLM))	0.8047	0.8150	0.8098	0.6021	0.7511
	Proposed Model (Group 1-CSS Type 2) (Stacking: 1st layer: LGBM, ADA, 2nd layer: RF, GNB, SVM2 Meta: LR)	**0.8119**	0.8204	**0.8161**	0.5340	0.7522
Group 2	Logistic Regression (LR)	0.8172	0.6952	0.7513	**0.7561**	0.7852
	XGBoost (XGB)	0.7938	0.7987	0.7962	0.4634	0.7736
	Random Forest (RF)	0.7903	0.7979	0.7941	0.4562	**0.7913**
	PyCaret (AdaBoost (ADA))	0.8058	0.8212	0.8134	0.6072	0.7996
	H2O.ai (Generalized Linear Modeling (GLM))	0.8116	0.8532	0.8318	0.6102	0.7629
	Proposed Model (Group 2-CSS Type 2) (Stacking: 1st layer: RF, Meta: LR)	**0.8303**	**0.8480**	**0.8391**	0.6105	0.7769
Group 3	Logistic Regression (LR)	0.8213	0.6815	0.7449	**0.7701**	0.7886
	XGBoost (XGB)	0.8018	0.8045	0.8031	0.4783	0.7812
	Light Gradient Boosting Machine (LGBM)	0.8078	0.8067	0.8072	0.5004	0.7950
	PyCaret (Logistic Regression (LR))	0.8073	0.8428	0.8247	0.6142	0.8003
	H2O.ai (Gradient Boosting Machine (GBM))	0.8069	0.8467	0.8263	0.6115	0.7721
	Proposed Model (Group 3-CSS Type 2) (Stacking: 1st layer: LGBM, SGD, Meta: LR)	**0.8628**	**0.8550**	**0.8589**	0.7585	**0.8187**
Group 4	Logistic Regression (LR)	0.8577	0.7665	0.8095	**0.8453**	0.8716
	XGBoost (XGB)	0.8714	0.8540	0.8626	0.7299	0.9057
	Light Gradient Boosting Machine (LGBM)	0.8834	0.8624	0.8728	0.7645	**0.9126**
	PyCaret (Gaussian Naïve Bayes (GNB))	0.8084	0.8248	0.8165	0.8193	0.8262
	H2O.ai (Gradient Boosting Machine (GBM))	0.8191	0.8590	0.8386	0.8025	0.7931
	Proposed Model (Group 4-CSS Type 2) (Stacking: 1st layer: LGBM, RF, Meta: LR)	**0.9203**	**0.9169**	**0.9186**	0.8193	0.9031
Group 5	Logistic Regression (LR)	0.8624	0.7700	0.8136	**0.8331**	0.8654
	XGBoost (XGB)	0.8767	0.8598	0.8682	0.7157	0.9062
	Light Gradient Boosting Machine (LGBM)	0.8807	0.8611	0.8708	0.7425	0.9115
	PyCaret (CatBoost (CAT))	0.7936	0.8504	0.8210	0.8083	0.8295
	H2O.ai (Generalized Linear Modeling (GLM))	0.8139	0.8550	0.8339	0.8251	0.8084
	Proposed Model (Group 5-CSS Type 2) (Stacking: 1st layer: LGBM, DT, SGD, Meta: LR)	**0.9330**	**0.9307**	**0.9318**	0.8269	**0.9121**

In Groups 1–3, we observed an improvement in performance associated with an increase in the number of clinical variables. However, a significant improvement in performance was observed in Groups 4 and 5, which included blood test results. This indicates that clinical variables related to blood tests significantly affect classification performance. However, the two AutoML approaches, PyCaret and H2O.ai, showed only slight improvements in Groups 4 and 5. This is because PyCaret and H2O.ai select the optimal single classifier rather than constructing stacking ensemble models for each group. Additionally, these two AutoML approaches typically streamline the parameter tuning process and utilize a limited selection of single classifiers to enhance user convenience. However,

these approaches limit the capacity for detailed hyperparameter tuning and restrict the use of a diverse range of models to adequately capture intricate patterns. As a result, they prevent the AutoML tools from fully understanding the complexities and specific interdependencies present in complex clinical datasets. Consequently, this could lead to the unique requirements or the most effective configurations that are essential for particular clinical tasks being overlooked.

Moreover, Logistic Regression shows high specificity across all groups but has a comparatively lower F1 score than other models, indicating a limitation in handling multiclass classifications with higher complexity. Overall, the proposed model outperformed single models, such as Logistic Regression, XGBoost, and LGBM, in most groups. Specifically, Group 5 achieved an F1 score of 0.9318 and AUC score of 0.9121, indicating good performance. These results demonstrated the potential advantages of the ensemble approach compared to single classifiers and AutoML for complex clinical classification tasks.

Table 9 presents the average performance results of the proposed adaptive stacking ensemble model in comparison to other single classifiers and AutoML approaches for predicting Type 2 severity across all groups.

Table 9. Average performance results of Type 3 severity for all groups.

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Average	Logistic Regression (LR)	0.7529	0.5748	0.6526	**0.7679**	0.7303
	XGBoost (XGB)	0.7292	0.7501	0.7409	0.5108	0.7554
	PyCaret	0.7249	0.7756	0.7493	0.6558	0.7031
	H2O.ai	0.6965	0.7849	0.7389	0.6167	0.6888
	Average of All Single Classifiers and AutoML	0.7206	0.6601	0.6787	0.5978	0.6812
	Proposed Model (CSS Type 3)	**0.7789**	**0.8183**	**0.7981**	0.6348	**0.7736**

The prediction of Type 3 severity requires a more detailed classification than that required for Type 2 severity. Type 3 is classified into five classes: '*asymptomatic*', '*mild*', '*moderate*', '*severe*', and '*fatal*'. The classification of the severity classes for Type 3 increased in complexity when compared with those for Types 1 and 2. As a result, our model's average F1 score decreased by 18.8% and 10.2% when compared with the results for Types 1 and 2, respectively. Similarly, the average AUC scores decreased by 16.4% and 5.9%, respectively. However, this reduction was also observed across other single classifiers and AutoML approaches. The proposed model achieved an F1 score of 0.7981 and an AUC score of 0.7736, outperforming the average of other models, including Logistic Regression, XGBoost, and two AutoML approaches.

Table 10 provides a summary of the comparative performance results for Type 3 severity across Groups 1–5. Comprehensive performance results of the comparisons with all 14 individual classifiers and AutoML approaches can be found in Table A3 of Appendix A.

Table 10. Summary of performance results of Type 3 severity for Groups 1–5.

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Group 1	Logistic Regression (LR)	**0.7172**	0.5643	0.6316	**0.6878**	0.6396
	XGBoost (XGB)	0.6709	0.7050	0.6875	0.3913	0.6458
	Light Gradient Boosting Machine (LGBM)	0.6825	0.7098	0.6959	0.4239	0.6568
	PyCaret (Logistic Regression (LR))	0.7109	0.7509	0.7304	0.6571	0.6461
	H2O.ai (Stacking: 1st layer: 5 GBM, 1 GLM, 2 DRF, Meta: GLM)	0.6849	0.7907	0.7340	0.4725	0.5425
	Proposed Model (Group 1-CSS Type 3)(Stacking: 1st layer: LGBM, GNB, 2nd layer: MLP, LR, Meta: LR)	0.7157	**0.7691**	**0.7414**	0.5333	**0.6584**

Table 10. *Cont.*

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Group 2	Logistic Regression (LR)	0.7299	0.5263	0.6116	**0.7353**	0.6900
	XGBoost (XGB)	0.6957	0.7253	0.7102	0.4394	0.6972
	Light Gradient Boosting Machine (LGBM)	0.7013	0.7306	0.7157	0.4510	**0.7096**
	PyCaret (Gaussian Naïve Bayes (GNB))	0.7328	0.7717	0.7517	0.6951	0.6897
	H2O.ai (Stacking: 1st layer: 22 GBM, 1 GLM, 2 DRF, Meta: GLM)	0.7001	0.7654	0.7313	0.4936	0.6975
	Proposed Model (Group 2-CSS Type 3)(Stacking: 1st layer: LGBM, RF, ADA, 2nd layer: MLP, LR, Meta: LR)	**0.7420**	**0.7749**	**0.7581**	0.5805	0.6914
Group 3	Logistic Regression (LR)	0.7386	0.5343	0.6251	**0.7472**	0.7017
	XGBoost (XGB)	0.7025	0.7297	0.7234	0.4512	0.7024
	Light Gradient Boosting Machine (LGBM)	0.7105	0.7368	0.7225	0.4549	0.7084
	PyCaret (Random Forest (RF))	0.7231	0.7897	0.7549	0.4928	0.7128
	H2O.ai (Generalized Linear Modeling (GLM))	0.6954	0.7825	0.7408	0.7036	0.7367
	Proposed Model (Group 3-CSS Type 3)(Stacking: 1st layer: LGBM, RF, ADA, 2nd layer: MLP, LR, Meta: LR)	**0.7566**	**0.7892**	**0.7726**	0.5906	**0.7355**
Group 4	Logistic Regression (LR)	0.7899	0.6249	0.6978	**0.8429**	0.8122
	XGBoost (XGB)	0.7915	0.7948	0.7931	0.6450	0.8660
	Light Gradient Boosting Machine (LGBM)	0.7916	0.7917	0.7916	0.6607	0.8744
	PyCaret (Gaussian Naïve Bayes (GNB))	0.7346	0.7775	0.7554	0.7558	0.7305
	H2O.ai (Stacking: 1st layer: 22 GBM, 1 GLM, 2 DRF, Meta: GLM)	0.6965	0.7966	0.7432	0.7214	0.7484
	Proposed Model (Group 4-CSS Type 3)(Stacking: 1st layer: LGBM, LR, 2nd layer: MLP, Meta: LR)	**0.8239**	**0.8631**	**0.8430**	0.7626	**0.8995**
Group 5	Logistic Regression (LR)	0.7887	0.6240	0.6967	**0.8264**	0.8079
	XGBoost (XGB)	0.7855	0.7956	0.7905	0.6272	0.8658
	Light Gradient Boosting Machine (LGBM)	0.7854	0.7925	0.7889	0.6363	0.8720
	PyCaret (CatBoost (CAT))	0.7231	0.7883	0.7543	0.6782	0.7362
	H2O.ai (Gradient Boosting Machine (GBM))	0.7054	0.7895	0.7451	0.6926	0.7187
	Proposed Model (Group 5-CSS Type 3)(Stacking: 1st layer: LGBM, SVM2, 2nd layer: MLP, LR, Meta: LR)	**0.8564**	**0.8952**	**0.8754**	0.7068	**0.9115**

Similar to the performance observed for Types 1 and 2, an increase in the number of clinical variables from Groups 1–5 resulted in improved F1 and AUC scores. However, the improvement in our proposed model from Group 2 to Group 3 was limited, with only a 1.5% increase in the F1 score and a 4.4% increase in the AUC score. Similarly, the performance of other single classifiers and AutoML approaches was observed to be lower in comparison. This suggests that a patient's current or past medical history may not be as crucial as significant predictive variables when patient symptom information is available. Groups 4 and 5, which included the blood test results as clinical variables, showed significant performance improvements. Thus, all experiments confirmed that blood test results could be used as important variables for predicting the severity of COVID-19.

Moreover, while the AutoML approaches PyCaret and H2O.ai demonstrated performance enhancements in Groups 1 and 2, they did not consistently outperform our proposed model. Specifically, PyCaret showed notable F1 scores, such as 0.7304 in Group 1 and 0.7517 in Group 2, by utilizing various single classifiers, including Logistic Regression and Gaussian Naïve Bayes. Similarly, H2O.ai, through its ensemble strategy that incorpo-

rates multiple models in stacking configurations, presented competitive results, with F1 scores of 0.7340 in Group 1 and 0.7313 in Group 2. However, despite these achievements, both AutoML approaches exhibited only slight enhancements in Groups 4 and 5. This issue becomes particularly apparent in scenarios requiring finer parameter calibration and model tuning due to the increased complexity introduced by a larger number of clinical variables. Specifically, H2O.ai's optimization focuses on model parameters, without directly optimizing the data itself, making its performance highly dependent on the characteristics of the data. This leads to potential shortcomings in effectively capturing the intricate patterns present in blood tests and other clinical variables.

Similar to the classification of severity in Type 2, Logistic Regression showed higher specificity but lower F1 and AUC scores than our proposed model. This implies that logistic regression has a low prediction rate for true positives and high number of false negatives for a specific class. In other words, it often misclassifies the severity classes, potentially leading to incorrect negative results. Therefore, the use of logistic regression tends to critical severity classes being overlooked, indicating that essential severity information that is necessary for treatment might be missed.

When predicting Type 3 severity, the proposed model in Group 5 showed the highest performance, with an F1 score of 0.8754 and AUC score of 0.9115. However, the specificity was observed to be relatively low. This means that the model accurately identified almost all individuals with severe disease, but also incorrectly classified a significant number of healthy individuals. Such a situation often occurs during the early diagnosis of serious diseases that can have drastic consequences if undiagnosed. Therefore, our proposed model for early diagnosis may misclassify some healthy individuals as having critical severity. However, it is crucial that the model does not overlook or miss actual severe cases. Therefore, prioritizing high F1 and AUC scores is an important strategy when making critical medical decisions. Accordingly, our results can play a critical role in making important diagnostic decisions in life-threatening situations.

Figure 10 illustrates a comparison of the performance between Logistic Regression, XGBoost, PyCaret, H2O.ai, and our proposed model, specifically focusing on F1 and AUC scores for Groups 1, 3, and 5 within our data-driven strategy across all data types. Based on these experiments, our observations can be summarized as follows:

(i) As the number of classified classes (i.e., types of severity) increases, the need for detailed classification leads to increased complexity, which, in turn, degrades the performance of each model. However, our proposed model demonstrated a superior performance compared to existing single models and AutoML approaches, providing better adaptive predictive results even when not all clinical variables could be collected due to the hospital's circumstances.

(ii) By comparing Groups 2 and 3, it was found that the patient's current or past medical history was not an essential clinical variable when information about patient symptoms is available. Therefore, an appropriate classifier performance can be achieved using the patient's personal information and symptom details.

(iii) The performance improves as the number of clinical variables increases. Clinical variables related to blood tests are particularly important for severity classification. However, AutoML approaches are constrained by their limited scope for detailed hyperparameter tuning and the restriction on the variety of models they can utilize. This can lead to a relative decrease in performance when dealing with complex clinical datasets with a larger number of clinical variables.

(iv) Our proposed stacking ensemble classifier consistently exhibited a superior performance to various types of single classifiers in all cases.

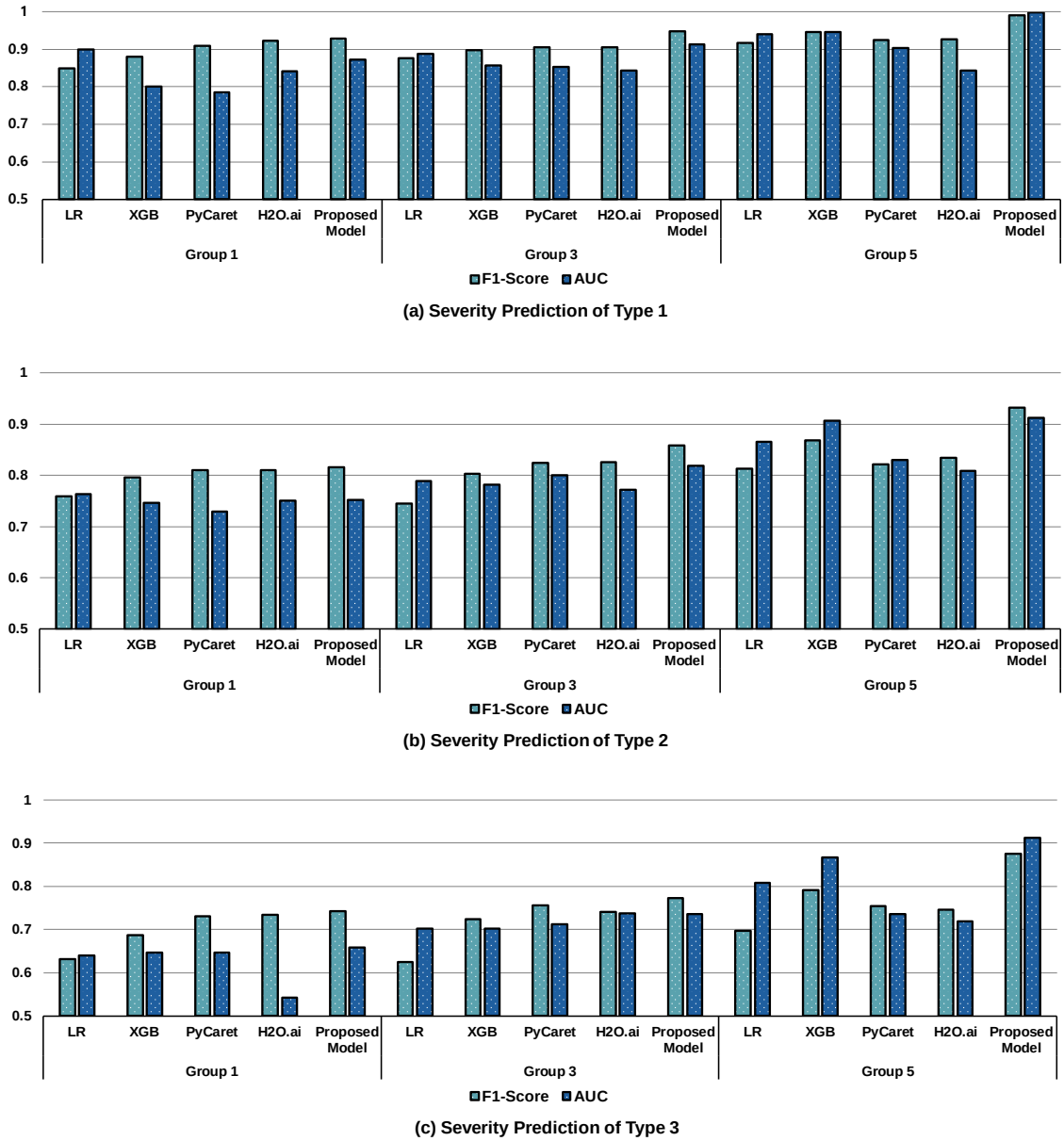


Figure 10. Performance comparison of Logistic Regression, XGBoost, PyCaret, H2O.ai, and our proposed model.

4.3. Effectiveness of Data-Driven Strategy and Oversampling

To evaluate the effectiveness of the data-driven strategy, we evaluated the performances of the original eight classes, each sub-classified into eight severity classes based on the *CSS*, and compared them with those of Types 1, 2, and 3, which were developed to address data imbalance issues. In this experiment, we compared the clinical variables of Group 5, which included blood test results, by utilizing 37 clinical variables. The performance of the original eight classes was measured using the LGBM, which was the most effective of the single classifiers. We evaluated the performances of Types 1, 2, and 3 using the proposed model.

Table 11 shows the precision, recall, F1 score, specificity, and AUC score for the clinical severities of the eight original classes obtained using the LGBM. Table 12 presents the precision, recall, F1 score, specificity, and AUC score for the clinical severity of Types 1, 2, and 3 using the proposed model.

Table 11. Performance results of the original eight classes using the LGBM model.

Type	Class Name	Precision	Recall	F1 Score	Specificity	AUC	Support
LGBM (8 Classes, Group 5)	No limitation of activity	0.8725	0.9855	0.9256	0.7570	0.9025	1337 (78.8%)
	Limited activity but no oxygen supply required	0.3456	0.4981	0.4081	0.8385	0.6715	101 (6%)
	Oxygen supply with nasal prong required	0.4480	0.3444	0.3894	0.9088	0.8147	143 (8.4%)
	Oxygen supply with facial mask required	0.0022	0.0011	0.0015	0.9894	0.8793	13 (0.8%)
	Non-invasive mechanical ventilation	0.0069	0.0067	0.0068	0.9802	0.7369	11 (0.6%)
	Invasive mechanical ventilation	0.0000	0.0000	0.0000	0.9973	0.9298	13 (0.8%)
	Multi-organ failure or ECMO	0.0000	0.0000	0.0000	0.9991	0.7367	4 (0.2%)
	Death	0.7984	0.7967	0.7975	0.9574	0.9713	74 (4.4%)
	Average/total Support	**0.2717**	**0.3291**	**0.3161**	**0.9285**	**0.8303**	1696 (100%)

Table 12. Performance results of clinical severity for Types 1, 2, and 3 using the proposed models.

Type	Class Name	Precision	Recall	F1 Score	Specificit	AUC	Support
Proposed Model (Type 1, Group 5)	Low Severity	0.9913	0.9955	0.9933	0.9517	0.9977	1594 (94%)
	High Severity	0.9888	0.9861	0.9875	0.9369	0.9977	102 (6%)
	Average/total support	**0.9901**	**0.9908**	**0.9904**	**0.9443**	**0.9977**	1696 (100%)
Proposed Model (Type 2, Group 5)	Week	0.9785	0.9873	0.9829	0.7456	0.9583	1438 (84.8%)
	Severe	0.8889	0.8498	0.8690	0.9823	0.7941	156 (9.2%)
	Critical	0.9317	0.9551	0.9434	0.7527	0.9839	102 (6.0%)
	Average/total support	**0.9330**	**0.9307**	**0.9318**	**0.8269**	**0.9121**	1696 (100%)
Proposed Model (Type 3, Group 5)	Asymptomatic	0.9723	0.9872	0.9804	0.7612	0.9044	1227 (72.3%)
	Mild	0.7951	0.8030	0.7993	0.5724	0.7114	101 (6.0%)
	Moderate	0.9223	0.9696	0.9473	0.7923	0.9046	156 (9.2%)
	Severe	0.7144	0.8338	0.7598	0.9629	0.9169	27 (1.6%)
	Fatal	0.8781	0.8824	0.8902	0.7823	0.9797	75 (4.4%)
	Average/total support	**0.8564**	**0.8952**	**0.8754**	**0.7742**	**0.8834**	1696 (100%)

The prediction results for the *CSS* of the original eight classes were generally poorer than those of the proposed model. This difference was primarily due to data imbalances, which were particularly noticeable in classes with a limited number of data (i.e., number of supports) in the test dataset. As a result, in some classes, such as '*Oxygen supply with facial mask required*', '*Non-invasive mechanical ventilation*', '*Invasive mechanical ventilation*', and '*Multi-organ failure or ECMO*', the Precision, Recall, and F1 scores were very low or close to zero. This indicates that the performance and reliability may be significantly reduced, especially in classes with insufficient data samples. The proposed model applies a data-driven strategy, reconfiguring the original eight classes into Types 1, 2, and 3. This adjustment alleviated the prevailing data imbalance and facilitated a relatively consistent performance across the different classes.

In addition, the use of an oversampling strategy effectively mitigated the shortcomings of the training data, thereby improving the overall performance of the model. Healthcare data have unique characteristics that can lead to concerns regarding performance degradation or the distortion of data attributes during oversampling. Therefore, we evaluated the performances of Logistic Regression, XGBoost, PyCaret, H2O.ai, and our proposed model by using Group 5 clinical variables across clinical severity types 1, 2, and 3. Evaluations were conducted both before and after applying oversampling using the SMOTE and ADASYN techniques. We utilized external libraries such as '*imbalanced-learn*' in Python to perform the oversampling and then processed the data under the same conditions. For each model included in the performance evaluation, a total of 20 experiments were conducted, and the average performance results were recorded. Table 13 and Figure 11 present the performance results of the proposed models, demonstrating the effectiveness of the oversampling strategy.

Table 13. Performance results before and after applying oversampling.

Type	Model	Before Oversampling		After Oversampling			
				SMOTE		ADASYN	
		F1 Score	AUC	F1 Score	AUC	F1 Score	AUC
Type 1 (Group 5)	Logistic Regression (LR)	0.8739	0.9128	0.9059	0.9562	0.9166	0.9407
	XGBoost (XGB)	0.9182	0.9163	0.9340	0.9348	0.9454	0.9460
	PyCaret (CatBoost (CAT))	0.9071	0.8962	0.9183	0.9011	0.9246	0.9024
	H2O.ai (Stacking: 1st layer: 1 GBM, 1 GLM, 2 DRF, Meta: GLM)	0.8626	0.8126	0.9031	0.8387	0.9267	0.8433
	Proposed Model (Group 5-CSS Type 1) (Stacking: 1st layer: RF, MLP, SGD 2nd layer: RF, SGD, Meta: LR)	**0.9315**	**0.9336**	**0.9623**	**0.9853**	**0.9904**	**0.9877**
Type 2 (Group 5)	Logistic Regression (LR)	0.7734	0.8316	0.8046	0.8599	0.8136	0.8654
	XGBoost (XGB)	0.8590	0.8925	0.8594	0.9052	0.8682	0.9062
	PyCaret (CatBoost (CAT))	0.8192	0.8214	0.8199	0.8214	0.8210	0.8295
	H2O.ai (Generalized Linear Modeling (GLM))	0.8013	0.7928	0.8312	0.8045	0.8339	0.8084
	Proposed Model (Group 5-CSS Type 2) (Stacking: 1st layer: LGBM, DT, SGD, Meta: LR)	**0.8972**	**0.8957**	**0.9158**	**0.9112**	**0.9318**	**0.9121**
Type 3 (Group 5)	Logistic Regression (LR)	0.6211	0.7724	0.6514	0.7935	0.6967	0.8079
	XGBoost (XGB)	0.7882	0.8325	0.7892	0.8527	0.7905	0.8658
	PyCaret (CatBoost (CAT))	0.7495	0.7312	0.7503	0.7341	0.7543	0.7362
	H2O.ai (Gradient Boosting Machine (GBM))	0.7366	0.7097	0.7399	0.7113	0.7451	0.7187
	Proposed Model (Group 5-CSS Type 3) (Stacking: 1st layer: LGBM, SVM2, 2nd layer: MLP, LR, Meta: LR)	**0.7991**	**0.8572**	**0.8524**	**0.8811**	**0.8754**	**0.9115**

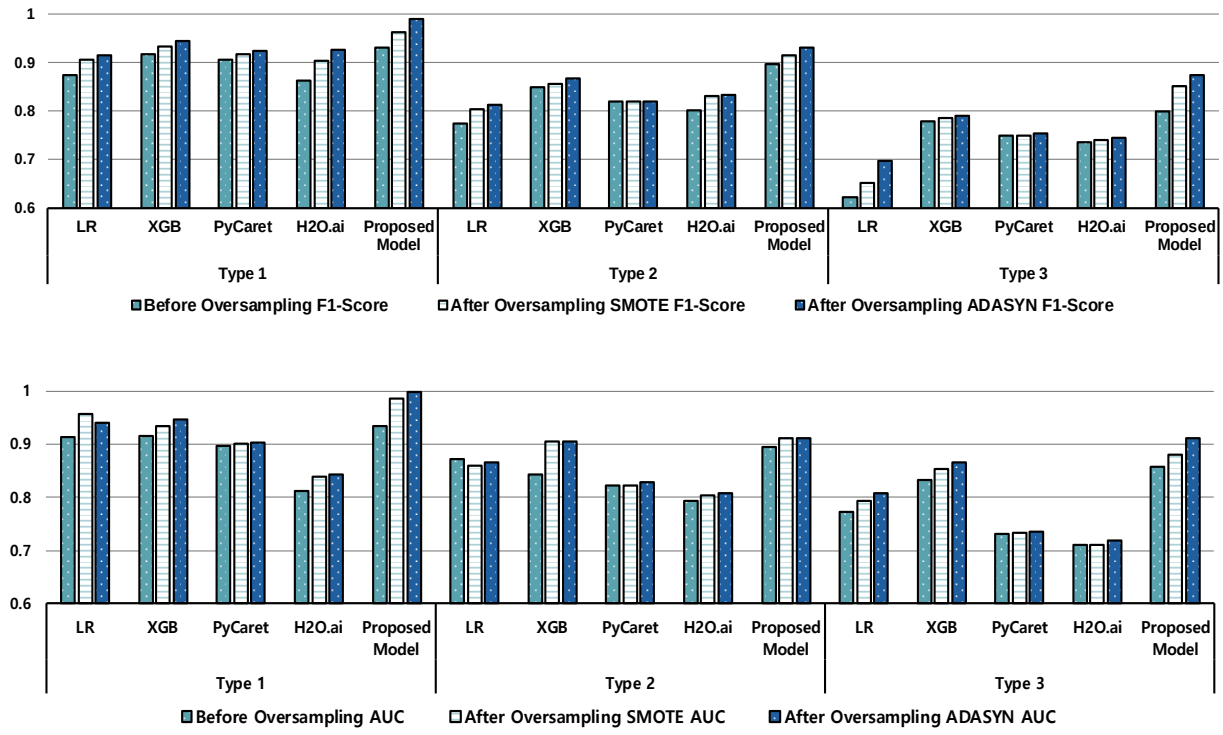


Figure 11. Comparison of performances before and after applying oversampling.

Overall, the oversampling technique contributed to the improvement in the F1 and AUC scores for all classifiers. For Type 1, the proposed model showed an increase in the F1 score of 3.08% when applying SMOTE and of 5.89% when applying ADASYN. Additionally,

the AUC score improved by 5.17% with SMOTE and 5.41% with ADASYN. For Type 2, the improvements were more moderate than those for Type 1. The F1 score increased by 1.86% for SMOTE and 3.46% for ADASYN, whereas the AUC score increased by 1.55% for SMOTE and 1.64% for ADASYN. Finally, for Type 3, there was a notable increase in the F1 score by 5.33% with SMOTE and 7.63% with ADASYN, and the AUC score increased by 3.39% and 6.43%, respectively.

In the case of XGBoost, PyCaret (CatBoost), and H2O.ai (GBM), oversampling did not lead to significant improvements across any of the types. This lack of improvement was attributed to the nature of the boosting algorithms. For instance, XGBoost and CatBoost construct new trees by correcting the errors in the previously created, weaker decision trees. Due to this approach, the oversampled data did not significantly contribute to the results. Consequently, the models in Type 2, especially those using boosting ensemble models such as LGBM and RF, as well as those utilizing AutoML approaches based on boosting techniques, did not show substantial performance improvements after oversampling. However, the proposed combination of models in Type 3, which included a variety of models such as MLP and SVM, demonstrated the most remarkable performance improvement.

4.4. Effect of Feature Selection Algorithm on Performance Results

To select the important features that influence clinical severity, we evaluated the effects of 37 input clinical variables on the severity and selected an optimal subset of features that could improve the prediction performance by removing less relevant features. For this purpose, we proposed a feature selection algorithm based on greedy search algorithms, including forward selection, backward elimination, and RFECV. To evaluate the effectiveness of our proposed feature selection algorithm, we conducted comparative experiments with no feature selection and with forward selection, backward elimination, RFECV, or the proposed algorithm.

Table 14 presents the features selected by each of the feature selection methods for Groups 1, 3, and 5, which were defined in the data-driven strategy for the 37 clinical variables. The detailed results for all groups are presented in Table A4 (Appendix B).

Table 14. Selected features from feature selection algorithm for Groups 1, 3, and 5.

Groups	Feature Selection	Selected Feature (Clinical Variables)
Group 1	None (9)	AGE, SEX, PREG, PREGW, BMI, SBP, DBP, HRI, TEMPI
	Forward Selection (5)	AGE, PREG, BMI, HRI, TEMPI
	Backward Elimination (5)	AGE, PREGW, SBP, HRI, TEMPI
	RFECV (5)	AGE, BMI, SBP, HRI, TEMPI
	Proposed Algorithm (5)	AGE, BMI, SBP, HRI, TEMPI
Group 3	None (32)	AGE, SEX, PREG, PREGW, BMI, SBP, DBP, HRI, TEMPI, FEVER, COUGH, SPUTUM, ST, RNR, MAM, FM, SOB, HEADA, ACC, VN, DIARR, DM, HTN, HF, CCD, ASTHMA, COPD, CKD, MALIG, CLD, RDAD, DEMEN
	Forward Selection (9)	AGE, TEMPI, FEVER, FM, SOB, ACC, HTN, RDAD, DEMEN
	Backward Elimination (28)	AGE, SEX, PREG, PREGW, SBP, HRI, TEMPI, FEVER, COUGH, SPUTUM, ST, RNR, MAM, FM, SOB, HEADA, ACC, DIARR, DM, HTN, CCD, ASTHMA, COPD, CKD, MALIG, CLD, DEMEN
	RFECV (26)	AGE, SEX, BMI, SBP, DBP, HRI, TEMPI, FEVER, COUGH, SPUTUM, ST, RNR, MAM, FM, SOB, HEADA, ACC, VN, DIARR, DM, HTN, CCD, ASTHMA, MALIG, DEMEN
	Proposed Algorithm (22)	AGE, SEX, SBP, HRI, TEMPI, FEVER, COUGH, SPUTUM, ST, RNR, MAM, FM, SOB, HEADA, ACC, DIARR, DM, HTN, CCD, ASTHMA, MALIG, DEMEN

Table 14. *Cont.*

Groups	Feature Selection	Selected Feature (Clinical Variables)
Group 5	None (37)	AGE, SEX, PREG, PREGW, BMI, SBP, DBP, HRI, TEMPI, FEVER, COUGH, SPUTUM, ST, RNR, MAM, FM, SOB, HEADA, ACC, VN, DIARR, DM, HTN, HF, CCD, ASTHMA, COPD, CKD, MALIG, CLD, RDAD, DEMEN, HGB, HCT, LYMPHO, PLT, WBC
	Forward Selection (13)	AGE, TEMPI, ST, MAM, SOB, ACC, DM, DEMEN, HGB, HCT, LYMPHO, PLT, WBC
	Backward Elimination (32)	AGE, SEX, PREG, PREGW, BMI, SBP, DBP, TEMPI, FEVER, COUGH, SPUTUM, ST, RNR, MAM, FM, SOB, HEADA, ACC, DM, HTN, HF, ASTHMA, COPD, CKD, MALIG, RDAD, DEMEN, HGB, HCT, LYMPHO, PLT, WBC
	RFECV (22)	AGE, SEX, BMI, TEMPI, FEVER, ST, SOB, HEADA, ACC, VN, DIARR, DM, ASTHMA, COPD, MALIG, CLD, DEMEN, HGB, HCT, LYMPHO, PLT, WBC
	Proposed Algorithm (19)	AGE, SEX, BMI, TEMPI, FEVER, ST, MAM, SOB, HEADA, ACC, DM, HTN, COPD, MALIG, DEMEN, HGB, HCT, LYMPHO, PLT, WBC

Each feature selection method selects a different number of features, except for Group 1, which has a smaller total number of features. The forward selection method begins without any selected features and incrementally adds features that contribute to performance improvements. Consequently, this method selects a relatively small number of features across all groups, thereby increasing the risk of overall performance degradation due to the inclusion of a few features. In contrast, the backward elimination method begins by including all features and sequentially removes those with the least effect on the performance. This method tends to consider most features as important, and therefore eliminates only a small number of features across all groups. As a result, it selects a larger number of features for all groups than the other methods. However, this method can lead to inefficient feature selection because an insufficient number of features are properly removed. The RFECV method operates by iteratively training the model and removing less-significant features. With each iteration, the model is retrained with the remaining features and less important features are discarded. This process continues until the optimal number of features is determined through cross-validation; thus, a greater number of features is removed from all groups when compared with those removed in backward elimination. Finally, the proposed algorithm combines the features extracted by forward selection, backward elimination, and RFECV, ultimately returning only the important features selected by at least two methods. This approach has the advantage of encompassing features that might have been overlooked by other feature selection algorithms.

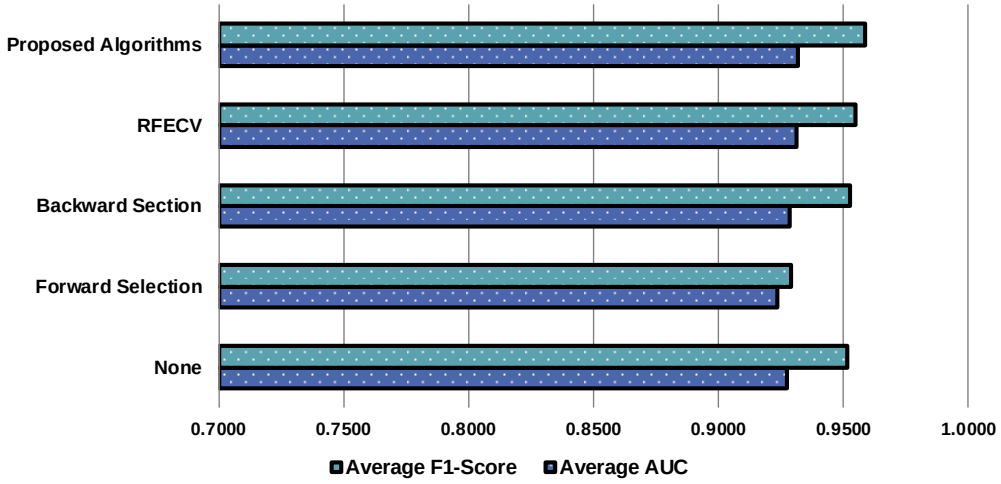
Table 15 shows the performance results of the feature selection for each group in Type 1 in terms of the F1 and AUC scores. To assess the feature-selection performance, we evaluated the performances of various feature-selection algorithms when applied to the proposed models derived for each type. The bold and underlined results in the table represent the most outstanding performances.

The results for Type 1, which included simple binary classification, generally showed high performance in terms of both F1 and AUC scores. Most algorithms exhibited the highest F1 and AUC scores in Group 5. In terms of the F1 score, the proposed algorithm showed a superior performance across all groups when compared with the other algorithms. Regarding the AUC, the other algorithms showed a competitive performance in Groups 2 and 3, but there was no significant difference when compared with the proposed algorithm. In Group 1, both RFECV and the proposed algorithm resulted in identical feature selection outcomes, leading to identical F1 and AUC scores of 0.9281 and 0.8722, respectively. For Type 1, there was no significant difference in performance because the number and types

of features selected by the algorithms were similar. Figure 12 illustrates the average F1 and AUC scores for each feature selection algorithm across Groups 1–5 for Type 1.

Table 15. Performance results of feature selection for each group in Type 1.

Feature Selection Algorithm	Group 1		Group 2		Group 3		Group 4		Group 5	
	F1-Score	AUC	F1-Score	AUC	F1-Score	AUC	F1-Score	AUC	F1-Score	AUC
None	0.9069	0.8542	0.9398	0.9081	0.9374	0.9115	0.9865	0.9772	0.9867	0.9867
Forward Selection	0.8957	0.8703	0.8796	0.8891	0.8973	0.8948	0.9860	0.9765	0.9858	0.9865
Backward Elimination	0.9252	0.8542	0.9317	**0.9115**	0.9318	0.9127	0.9863	0.9770	0.9873	0.9870
RFECV	**0.9281**	**0.8722**	0.9288	0.8999	0.9430	**0.9215**	0.9865	0.9771	0.9874	0.9863
Proposed Algorithm	**0.9281**	**0.8722**	**0.9414**	0.9098	**0.9472**	0.9129	**0.9871**	**0.9774**	**0.9904**	**0.9877**



	None	Forward Selection	Backward Elimination	RFECV	Proposed Algorithm
Average F1-Score	0.9515	0.9289	0.9525	0.9548	0.9588
Average AUC	0.9275	0.9234	0.9285	0.9314	0.9320

Figure 12. Average F1 and AUC scores of the feature selection algorithm across Groups 1–5 for Type 1.

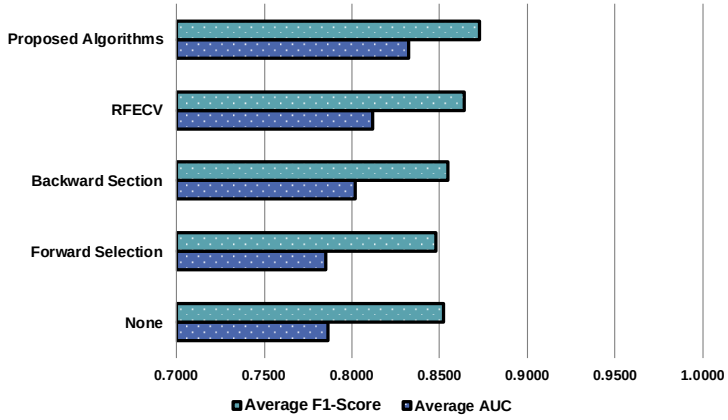
Tables 16 and 17 present the performance results of the feature selection for each group for Types 2 and 3 in terms of the F1 and AUC scores. It was observed that, for Types 2 and 3, which had a more diverse range of classes for classification, there was an overall decrease in performance in terms of both F1 and AUC scores when compared with the result for Type 1. Figures 13 and 14 illustrate the average F1 and AUC scores for each feature selection algorithm across Groups 1–5 for Types 2 and 3, respectively.

Table 16. Performance results of feature selection for each group for Type 2.

Feature Selection Algorithm	Group 1		Group 2		Group 3		Group 4		Group 5	
	F1-Score	AUC	F1-Score	AUC	F1-Score	AUC	F1-Score	AUC	F1-Score	AUC
None	0.7957	0.7034	0.8081	0.7340	0.8378	0.7366	0.9099	0.8537	0.9086	0.8944
Forward Selection	0.8035	0.6974	0.7853	0.7356	0.8366	0.7285	0.9079	0.8621	0.9063	0.9023
Backward Elimination	0.8052	0.7112	0.8185	0.7486	0.8389	0.7826	0.9045	0.8627	0.9056	0.9031
RFECV	**0.8161**	**0.7522**	0.8233	0.7391	0.8429	0.7969	0.9091	0.8662	0.9281	0.9039
Proposed Algorithms	**0.8161**	**0.7522**	**0.8391**	**0.7769**	**0.8589**	**0.8187**	**0.9186**	**0.9031**	**0.9318**	**0.9121**

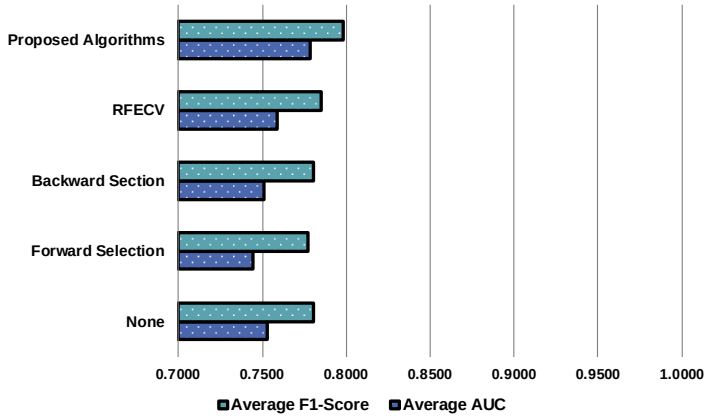
Table 17. Performance results of feature selection for each group for Type 3.

Feature Selection Algorithm	Group 1		Group 2		Group 3		Group 4		Group 5	
	F1-Score	AUC	F1-Score	AUC	F1-Score	AUC	F1-Score	AUC	F1-Score	AUC
None	0.7334	0.6379	0.7515	0.6853	0.7586	0.7233	0.8262	0.8266	0.8313	0.8901
Forward Selection	0.7345	0.6277	0.7541	0.6743	0.7622	0.7142	0.8285	0.8244	0.8356	0.8799
Backward Elimination	0.7379	0.6220	0.7488	0.6823	0.7488	0.7246	0.8295	0.8255	0.8373	0.9019
RFECV	**0.7414**	**0.6431**	0.7526	0.6890	0.7605	0.7285	0.8332	0.8309	0.8389	0.9043
Proposed Algorithms	**0.7414**	**0.6584**	**0.7581**	**0.6914**	**0.7726**	**0.7355**	**0.8430**	**0.8955**	**0.8754**	**0.9115**



	None	Forward Selection	Backward Elimination	RFECV	Proposed Algorithm
Average F1-Score	0.8520	0.8479	0.8545	0.8639	0.8729
Average AUC	0.7862	0.7852	0.8016	0.8117	0.8326

Figure 13. Average F1 and AUC scores for feature selection algorithm across groups 1–5 for Type 2.



	None	Forward Selection	Backward Elimination	RFECV	Proposed Algorithm
Average F1-Score	0.7802	0.7830	0.7805	0.7853	0.7981
Average AUC	0.7526	0.7441	0.7513	0.7592	0.7785

Figure 14. Average F1 and AUC scores for feature selection algorithm across groups 1–5 for Type 3.

In the case of forward selection, the F1 and AUC scores for Types 2 and 3 decreased when compared with that in the case where no feature selection was performed, mainly because the number of selected features was too small. In Type 2, when compared with the results obtained when no feature selection was performed, the average F1 score decreased by 0.41% and the AUC score decreased by 0.1%. In Type 3, the F1 score decreased by 0.32% and the AUC score decreased by 0.85% when compared with the results for the

case when no feature selection was performed. With backward elimination, there was no significant improvement in the performance when compared with the results obtained when no feature selection was performed. In Type 2, the average F1 score increased by 0.25% and the AUC score increased by 1.54%, whereas in Type 3, the average F1 score increased by 0.03% and the AUC score decreased by 0.14%. This was because backward elimination removes a relatively small number of features. The performance of RFECV was better than the performance of forward selection and backward elimination. This was because RFECV can remove or include important features through recursive model retraining during the feature selection process. In Type 2, RFECV showed an average increase of 1.19% in the F1 score and 2.54% in the AUC score when compared with the results obtained when no feature selection was performed. In Type 3, average increases of 0.83% in the F1 score and 1.51% in the AUC score were observed when compared with the results when no feature selection was performed. Finally, the proposed algorithm outperformed the other feature-selection algorithms. Type 2 demonstrated an average increase of 2.09% in the F1 score and 4.64% in the AUC score. Similarly, for Type 3, the proposed algorithm showed an average increase of 1.76% in the F1 score and 2.72% in the AUC score.

5. Conclusions

In this paper, we presented an adaptive stacking ensemble technique to effectively identify various COVID-19 severity levels in patients. This technique could be particularly useful in the early stages of the healthcare response to pandemics, such as COVID-19, even when resources are limited. To enhance the generalizability of our model, we utilized a nationwide dataset provided by the South Korean government, which included data on 5644 patients from more than 100 hospitals. Using a data-driven strategy, we grouped the clinical variables, developed adaptive models for various combinations, and presented the results for each group and severity type. We also analyzed important clinical variables using feature selection methods to improve our severity prediction model's performance by focusing on highly important features and removing variables of low relevance. Finally, we proposed a method for automatically constructing an adaptive stacking ensemble model that predicts the severity levels more accurately than existing single classifiers and AutoML approaches.

Despite our contributions, our research faced several limitations. Stacking ensemble models typically face challenges such as computational complexity and an extensive computation time. Although our processes, such as data preprocessing, feature selection, and oversampling, are automated using our proposed algorithms, they still require significant computational resources. Consequently, rapidly generating these models remains challenging in time-sensitive tasks and during data updates. Furthermore, although our implementation outperformed AutoML approaches in experimental evaluations, we were unable to enhance computational efficiency compared to AutoML approaches, which often utilize cloud computing or parallel processing. Instead, our goal was to enhance the performance in the identification of various COVID-19 severity levels within the constraints of clinical environments. Moreover, since our research was based on data from South Korean patients, the results may not be generalizable to other ethnic groups, such as Caucasians or Middle Eastern Asians. Therefore, external validation using a more diverse population is necessary to evaluate the effectiveness of the model across different ethnic groups. Additionally, our model's performance needs to be verified in real-world scenarios. Our research was limited to clinical epidemiological datasets.

In future work, we aim to apply our technique to diverse datasets, including those from various ethnic groups. To achieve this, we plan to develop a user-friendly web application. This application will enable a real-time training framework that allows our model to learn from prospectively collected data and provides optimizations to reduce computational overhead via the use of cloud computing resources. Furthermore, the surge in COVID-19 cases has led to a significant increase in medical documents, such as

clinical notes, offering new opportunities for our approach to identify patient severity through NLP and LLMs. However, LLMs present their own challenges, including the risk of generating hallucinated content and a dependency on prompt engineering for quality results. We intend to address these challenges and incorporate LLMs into our technique. This will automate the interpretation of text-based unstructured data. This integration could significantly enhance the strengths of our data-driven strategy. Furthermore, we will examine the model's adaptability and relevance in the post-COVID period, aiming to enhance its practical implementation. Our goal is not only to refine the technical aspects of our approach but also to ensure its practical applicability in real-world healthcare settings.

Author Contributions: Conceptualization: G.-W.K., H.S. and D.-H.L.; Methodology: G.-W.K.; Software: G.-W.K. and C.-Y.J.; Validation: G.-W.K., H.S. and D.-H.L.; Formal Analysis: G.-W.K., H.S. and C.-Y.J.; Investigation: G.-W.K.; Resources: C.-Y.J., H.S. and D.-H.L.; Data Curation: G.-W.K., C.-Y.J. and H.S.; Writing—Original Draft Preparation: G.-W.K.; Writing—Review and Editing: G.-W.K., C.-Y.J. and D.-H.L.; Visualization: G.-W.K. and C.-Y.J.; Supervision: D.-H.L.; Project Administration: G.-W.K. and D.-H.L.; Funding Acquisition: G.-W.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education, Science and Technology (NRF-2021R1G1A1006381).

Institutional Review Board Statement: The protocol of this retrospective study was approved by the Institutional Review Board of Korea University Ansan Hospital (K2021-0013-005) and followed the principles of the Declaration of Helsinki.

Informed Consent Statement: The requirement for obtaining informed patient consent was waived by the institutional review board (K2021-0013-005) due to the retrospective nature of the study.

Data Availability Statement: The data supporting the findings of this paper were provided by the KCDC, but their availability is subject to certain restrictions. As these data were used under a license for this specific study, they are not publicly accessible. Due to KDCA guidelines that prohibit the sharing of data by researchers, it is not possible to share the data publicly. However, they can be made available by the authors upon reasonable request and with the permission of the KCDC.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Performance Results of the Comparison with All 14 Individual Classifiers and AutoML Approaches

Table A1. Detailed performance results of Type 1 severity for Groups 1–5.

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Group 1	Logistic Regression (LR)	**0.9416**	0.7740	0.8496	**0.7604**	**0.8989**
	Support Vector Machine 1 (SVM 1)	0.8902	0.7846	0.8341	0.7336	0.8503
	Support Vector Machine 2 (SVM 2)	0.8824	0.7895	0.8334	0.3898	0.7718
	Stochastic Gradient Descent (SGD)	0.9089	0.7766	0.8376	0.7262	0.8438
	K-Nearest Neighbor (KNN)	0.8814	0.8443	0.8625	0.4832	0.6513
	Multi-Layer Perceptron (MLP)	0.9087	0.7121	0.7985	0.7841	0.8505
	Gaussian Naïve Bayes (GNB)	0.9049	0.8306	0.8662	0.6058	0.8312
	Decision Tree (DT)	0.8835	0.8434	0.8630	0.5106	0.5930
	Extra Trees Classifier (ExTree)	0.8861	0.8394	0.8621	0.5517	0.6086
	XGBoost (XGB)	0.8883	0.8735	0.8808	0.4644	0.8007
	Light Gradient Boosting Machine (LGBM)	0.8932	0.8770	0.8850	0.5059	0.8105
	Random Forest (RF)	0.8859	0.8602	0.8729	0.4911	0.8097
	AdaBoost (ADA)	0.8922	0.8540	0.8697	0.4756	0.8445
	CatBoost (CAT)	0.8922	0.8757	0.8839	0.5540	0.8197

Table A1. *Cont.*

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Group 1	PyCaret (CatBoost (CAT))	0.9188	0.8997	0.9091	0.6092	0.7849
	H2O.ai (Gradient Boosting Machine (GBM))	0.9347	0.9101	0.9223	0.7278	0.8406
	Proposed Model (Group 1-CSS Type 1) (Stacking: 1st layer: LGBM, 2nd layer: XGB, MLP, Meta: LR)	0.9262	**0.9301**	**0.9281**	0.6026	0.8722
Group 2	Logistic Regression (LR)	**0.9479**	0.7939	0.8641	**0.8168**	**0.9327**
	Support Vector Machine 1 (SVM 1)	0.9155	0.7925	0.8496	0.7892	0.8799
	Support Vector Machine 2 (SVM 2)	0.8691	0.8671	0.8726	0.2650	0.7755
	Stochastic Gradient Descent (SGD)	0.9146	0.7585	0.8293	0.8076	0.8735
	K-Nearest Neighbor (KNN)	0.8962	0.9138	0.9049	0.6083	0.7117
	Multi-Layer Perceptron (MLP)	0.9105	0.8107	0.8577	0.7077	0.8616
	Gaussian Naïve Bayes (GNB)	0.9150	0.7996	0.8534	0.7758	0.8540
	Decision Tree (DT)	0.8944	0.8695	0.8818	0.5605	0.6250
	Extra Trees Classifier (ExTree)	0.8970	0.8589	0.8775	0.6355	0.6571
	XGBoost (XGB)	0.9040	0.8876	0.8957	0.5823	0.8495
	Light Gradient Boosting Machine (LGBM)	0.9055	0.8894	0.8964	0.5893	0.8562
	Random Forest (RF)	0.9006	0.8721	0.8861	0.5482	0.8626
	AdaBoost (ADA)	0.9014	0.8729	0.8869	0.5482	0.8739
	CatBoost (CAT)	0.9084	0.8847	0.8974	0.5759	0.8645
	PyCaret (Random Forest (RF))	0.8993	0.8933	0.8963	0.5423	0.8462
	H2O.ai (Stacking: 1st layer: 1 GBM, 1 GLM, 2 DRF, Meta: GLM)	0.9354	0.8592	0.8957	0.8096	0.8410
	Proposed Model (Group 2-CSS Type 1) (Stacking: 1st layer: CAT, 2nd layer: CAT, LR, Meta: LR)	0.9407	**0.9421**	**0.9414**	0.7214	0.9098
Group 3	Logistic Regression (LR)	**0.9497**	0.8129	0.8760	**0.8180**	0.8880
	Support Vector Machine 1 (SVM 1)	0.9185	0.8005	0.8554	0.8172	0.8851
	Support Vector Machine 2 (SVM 2)	0.8595	0.8744	0.8669	0.0167	0.7862
	Stochastic Gradient Descent (SGD)	0.9170	0.7903	0.8489	0.8097	0.8863
	K-Nearest Neighbor (KNN)	0.9029	0.8775	0.8900	0.4367	0.7164
	Multi-Layer Perceptron (MLP)	0.9100	0.8324	0.8695	0.6678	0.8549
	Gaussian Naïve Bayes (GNB)	0.9148	0.8138	0.8613	0.7561	0.8403
	Decision Tree (DT)	0.8949	0.8629	0.8786	0.5945	0.6385
	Extra Trees Classifier (ExTree)	0.8927	0.8567	0.8743	0.5872	0.6319
	XGBoost (XGB)	0.9054	0.8916	0.8984	0.5619	0.8578
	Light Gradient Boosting Machine (LGBM)	0.9008	0.8947	0.8977	0.6240	0.8698
	Random Forest (RF)	0.9098	0.8925	0.9011	0.5554	0.8654
	AdaBoost (ADA)	0.9098	0.8972	0.9035	0.5554	0.8777
	CatBoost (CAT)	0.9080	0.8943	0.9011	0.5758	0.8715
	PyCaret (CatBoost (CAT))	0.9032	0.9075	0.9053	0.5237	0.8532
	H2O.ai (Gradient Boosting Machine (GBM))	0.9358	0.8760	0.9049	0.8085	0.8426
	Proposed Model (Group 3-CSS Type 1) (Stacking: 1st layer: LGBM, CAT, RF, 2nd layer: LGBM, GNB, Meta: LR)	0.9471	**0.9474**	**0.9472**	0.6526	**0.9129**
Group 4	Logistic Regression (LR)	0.9363	0.8850	0.9099	0.8914	0.9412
	Support Vector Machine 1 (SVM 1)	0.9340	0.8744	0.9032	0.8907	0.9412
	Support Vector Machine 2 (SVM 2)	0.8533	0.8898	0.8712	0.0602	0.8965
	Stochastic Gradient Descent (SGD)	0.9354	0.8810	0.9074	0.8912	0.9401
	K-Nearest Neighbor (KNN)	0.9395	0.9199	0.9296	0.6803	0.8510
	Multi-Layer Perceptron (MLP)	0.9467	0.9182	0.9322	0.8798	0.9445

Table A1. *Cont.*

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Group 4	Gaussian Naïve Bayes (GNB)	0.9277	0.8536	0.8891	0.8550	0.9035
	Decision Tree (DT)	0.9491	0.9292	0.9390	0.7635	0.8564
	Extra Trees Classifier (ExTree)	0.9310	0.9075	0.9191	0.6726	0.8001
	XGBoost (XGB)	0.9573	0.9363	0.9467	0.8741	0.9469
	Light Gradient Boosting Machine (LGBM)	0.9565	0.9358	0.9460	0.8534	0.9476
	Random Forest (RF)	0.9566	0.9358	0.9461	0.8603	0.9474
	AdaBoost (ADA)	0.9566	0.9343	0.9453	0.8603	0.9478
	CatBoost (CAT)	0.9575	0.9343	0.9458	0.8741	0.9479
	PyCaret (Logistic Regression (LR))	0.9313	0.8997	0.9152	0.8852	0.9410
	H2O.ai (Stacking: 1st layer: 1 GBM, 1 GLM, 2 DRF, Meta: GLM)	0.9357	0.9136	0.9245	0.8685	0.8439
	Proposed Model (Group 4-CSS Type 1) (Stacking: 1st layer: RF, 2nd layer: RF, LR, Meta: LR)	**0.9871**	**0.9872**	**0.9871**	**0.8959**	**0.9974**
Group 5	Logistic Regression (LR)	0.9376	0.8965	0.9166	0.8646	0.9407
	Support Vector Machine 1 (SVM 1)	0.9358	0.8903	0.9125	0.8642	0.9405
	Support Vector Machine 2 (SVM 2)	0.8633	0.8908	0.8768	0.0602	0.8571
	Stochastic Gradient Descent (SGD)	0.9386	0.8996	0.9187	0.8648	0.9384
	K-Nearest Neighbor (KNN)	0.9322	0.9115	0.9217	0.6454	0.8145
	Multi-Layer Perceptron (MLP)	0.9455	0.9173	0.9312	0.8660	0.9420
	Gaussian Naïve Bayes (GNB)	0.9221	0.8447	0.8817	0.7994	0.8935
	Decision Tree (DT)	0.9500	0.9270	0.9384	0.8459	0.8965
	Extra Trees Classifier (ExTree)	0.9148	0.8938	0.9042	0.5066	0.7102
	XGBoost (XGB)	0.9561	0.9350	0.9454	0.8671	0.9460
	Light Gradient Boosting Machine (LGBM)	0.9558	0.9345	0.9450	0.8671	0.9473
	Random Forest (RF)	0.9588	0.9389	0.9487	0.8329	0.9467
	AdaBoost (ADA)	0.9588	0.9389	0.9487	0.8329	0.9469
	CatBoost (CAT)	0.9583	0.9372	0.9476	0.8879	0.9476
	PyCaret (CatBoost (CAT))	0.9176	0.9318	0.9246	0.8321	0.9024
	H2O.ai (Stacking: 1st layer: 1 GBM, 1 GLM, 2 DRF, Meta: GLM)	0.9355	0.9180	0.9267	0.8679	0.8433
	Proposed Model (Group 5-CSS Type 1) (Stacking: 1st layer: RF, MLP, SGD 2nd layer: RF, SGD, Meta: LR)	**0.9901**	**0.9908**	**0.9904**	**0.9443**	**0.9977**
Average	Logistic Regression (LR)	0.9426	0.8325	0.8832	0.8302	0.9203
	XGBoost (XGB)	0.9222	0.9048	0.9134	0.6700	0.8802
	PyCaret	0.9140	0.9064	0.9101	0.6785	0.8655
	H2O.ai	0.9354	0.8954	0.9148	0.8165	0.8423
	Average of 14 Single Classifiers	0.9168	0.8698	0.8920	0.6676	0.8487
	Average of 2 AutoML Approaches	0.9247	0.9009	0.9125	0.7475	0.8539
	Average of All Single Classifiers and AutoML	0.9178	0.8737	0.8946	0.6776	0.8494
	Proposed Model (CSS Type 1)	**0.9582**	**0.9595**	**0.9588**	**0.7634**	**0.9380**

Table A2. Detailed performance results of Type 2 severity for Groups 1–5.

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Group 1	Logistic Regression (LR)	0.8060	0.7186	0.7598	**0.7670**	**0.7632**
	Support Vector Machine 1 (SVM 1)	0.8031	0.7265	0.7629	0.6833	0.7562
	Support Vector Machine 2 (SVM 2)	0.7718	0.6659	0.7149	0.5858	0.6890
	Stochastic Gradient Descent (SGD)	0.7766	0.7341	0.7548	0.6144	0.6720
	K-Nearest Neighbor (KNN)	0.7769	0.7721	0.7745	0.5030	0.6243
	Multi-Layer Perceptron (MLP)	0.8133	0.6558	0.7261	0.7047	0.7590
	Gaussian Naïve Bayes (GNB)	0.7960	0.7447	0.7695	0.6366	0.7314
	Decision Tree (DT)	0.7629	0.7619	0.7624	0.4485	0.5652
	Extra Trees Classifier (ExTree)	0.7590	0.7416	0.7502	0.4641	0.5615
	XGBoost (XGB)	0.7690	**0.8243**	0.7957	0.4468	0.7470
	Light Gradient Boosting Machine (LGBM)	0.7783	0.8162	0.7968	0.5067	0.7569
	Random Forest (RF)	0.7699	0.7977	0.7836	0.4472	0.7469
	AdaBoost (ADA)	0.7699	0.7977	0.7836	0.4472	0.7086
	CatBoost (CAT)	0.7767	0.8163	0.7960	0.5172	0.7426
	PyCaret (AdaBoost (ADA))	0.7971	0.8226	0.8096	0.5954	0.7299
	H2O.ai (Generalized Linear Modeling (GLM))	0.8047	0.8150	0.8098	0.6021	0.7511
	Proposed Model (Group 1-CSS Type 2) (Stacking: 1st layer: LGBM, ADA, 2nd layer: RF, GNB, SVM2 Meta: LR)	**0.8119**	0.8204	**0.8161**	0.5340	0.7522
Group 2	Logistic Regression (LR)	0.8172	0.6952	0.7513	**0.7561**	0.7852
	Support Vector Machine 1 (SVM 1)	0.8226	0.7023	0.7577	0.7458	0.7861
	Support Vector Machine 2 (SVM 2)	0.7134	0.7620	0.7369	0.1542	0.7154
	Stochastic Gradient Descent (SGD)	0.8139	0.7218	0.7651	0.7074	0.7740
	K-Nearest Neighbor (KNN)	0.7790	0.7572	0.7679	0.4741	0.6349
	Multi-Layer Perceptron (MLP)	0.8186	0.6736	0.7391	0.7554	0.7662
	Gaussian Naïve Bayes (GNB)	0.8105	0.7059	0.7546	0.6960	0.7727
	Decision Tree (DT)	0.7720	0.7377	0.7545	0.4820	0.5773
	Extra Trees Classifier (ExTree)	0.7806	0.7253	0.7519	0.5345	0.6011
	XGBoost (XGB)	0.7938	0.7987	0.7962	0.4634	0.7736
	Light Gradient Boosting Machine (LGBM)	0.8007	0.8010	0.8008	0.4928	0.7890
	Random Forest (RF)	0.7903	0.7979	0.7941	0.4562	**0.7913**
	AdaBoost (ADA)	0.7898	0.7961	0.7938	0.4537	0.7663
	CatBoost (CAT)	0.7909	0.7890	0.7899	0.4820	0.7697
	PyCaret (AdaBoost (ADA))	0.8058	0.8212	0.8134	0.6072	0.7996
	H2O.ai (Generalized Linear Modeling (GLM))	0.8116	0.8532	0.8318	0.6102	0.7629
	Proposed Model (Group 2-CSS Type 2) (Stacking: 1st layer: RF, Meta: LR)	**0.8303**	**0.8480**	**0.8391**	0.6105	0.7769
Group 3	Logistic Regression (LR)	0.8213	0.6815	0.7449	**0.7701**	0.7886
	Support Vector Machine 1 (SVM 1)	0.8257	0.6873	0.7502	0.7687	0.7923
	Support Vector Machine 2 (SVM 2)	0.7038	0.7740	0.7372	0.1092	0.7231
	Stochastic Gradient Descent (SGD)	0.8202	0.6802	0.7437	0.7565	0.7780
	K-Nearest Neighbor (KNN)	0.7895	0.7448	0.7665	0.4365	0.6327
	Multi-Layer Perceptron (MLP)	0.8130	0.7041	0.7546	0.6997	0.7663
	Gaussian Naïve Bayes (GNB)	0.8507	0.1030	0.1838	0.7396	0.7589
	Decision Tree (DT)	0.7839	0.7443	0.7636	0.5162	0.5975
	Extra Trees Classifier (ExTree)	0.7669	0.7178	0.7415	0.4705	0.5748
	XGBoost (XGB)	0.8018	0.8045	0.8031	0.4783	0.7812
	Light Gradient Boosting Machine (LGBM)	0.8078	0.8067	0.8072	0.5004	0.7950

Table A2. *Cont.*

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Group 3	Random Forest (RF)	0.8043	0.8054	0.8048	0.4761	0.8010
	AdaBoost (ADA)	0.8043	0.8054	0.8048	0.4761	0.7319
	CatBoost (CAT)	0.7985	0.7943	0.7964	0.4992	0.7800
	PyCaret (Logistic Regression (LR))	0.8073	0.8428	0.8247	0.6142	0.8003
	H2O.ai (Gradient Boosting Machine (GBM))	0.8069	0.8467	0.8263	0.6115	0.7721
	Proposed Model (Group 3-CSS Type 2) (Stacking: 1st layer: LGBM, SGD, Meta: LR)	**0.8628**	**0.8550**	**0.8589**	0.7585	**0.8187**
Group 4	Logistic Regression (LR)	0.8577	0.7665	0.8095	**0.8453**	0.8716
	Support Vector Machine 1 (SVM 1)	0.8596	0.7576	0.8054	0.8345	0.8720
	Support Vector Machine 2 (SVM 2)	0.6891	0.7974	0.7393	0.0845	0.8003
	Stochastic Gradient Descent (SGD)	0.8471	0.7757	0.8098	0.8011	0.8352
	K-Nearest Neighbor (KNN)	0.8275	0.8023	0.8147	0.6232	0.7293
	Multi-Layer Perceptron (MLP)	0.8629	0.8089	0.8350	0.8057	0.8743
	Gaussian Naïve Bayes (GNB)	0.8768	0.1114	0.1977	0.8193	0.8127
	Decision Tree (DT)	0.8353	0.7961	0.8152	0.7075	0.6899
	Extra Trees Classifier (ExTree)	0.8175	0.7634	0.7895	0.6439	0.6722
	XGBoost (XGB)	0.8714	0.8540	0.8626	0.7299	0.9057
	Light Gradient Boosting Machine (LGBM)	0.8834	0.8624	0.8728	0.7645	0.9126
	Random Forest (RF)	0.8703	0.8492	0.8596	0.7365	0.9011
	AdaBoost (ADA)	0.8703	0.8492	0.8596	0.7365	0.7862
	CatBoost (CAT)	0.8739	0.8567	0.8652	0.7301	0.9021
	PyCaret (Gaussian Naïve Bayes (GNB))	0.8084	0.8248	0.8165	0.8193	0.8262
	H2O.ai (Gradient Boosting Machine (GBM))	0.8191	0.8590	0.8386	0.8025	0.7931
	Proposed Model (Group 4-CSS Type 2) (Stacking: 1st layer: LGBM, RF, Meta: LR)	**0.9203**	**0.9169**	**0.9186**	0.8683	**0.9031**
Group 5	Logistic Regression (LR)	0.8624	0.7700	0.8136	**0.8683**	0.8654
	Support Vector Machine 1 (SVM 1)	0.8645	0.7616	0.8098	0.8444	0.8661
	Support Vector Machine 2 (SVM 2)	0.6888	0.7974	0.7391	0.0821	0.7721
	Stochastic Gradient Descent (SGD)	0.8500	0.7607	0.8029	0.8135	0.8111
	K-Nearest Neighbor (KNN)	0.8130	0.7846	0.7985	0.5535	0.7180
	Multi-Layer Perceptron (MLP)	0.8548	0.8094	0.8315	0.7621	0.8753
	Gaussian Naïve Bayes (GNB)	0.8735	0.4325	0.5785	0.8233	0.8290
	Decision Tree (DT)	0.8537	0.8182	0.8356	0.7337	0.7369
	Extra Trees Classifier (ExTree)	0.8001	0.7536	0.7762	0.5801	0.6399
	XGBoost (XGB)	0.8767	0.8598	0.8682	0.7157	0.9062
	Light Gradient Boosting Machine (LGBM)	0.8807	0.8611	0.8708	0.7425	0.9115
	Random Forest (RF)	0.8715	0.8514	0.8613	0.7367	0.9036
	AdaBoost (ADA)	0.8715	0.8514	0.8613	0.7367	0.6562
	CatBoost (CAT)	0.8726	0.8554	0.8639	0.7276	0.9038
	PyCaret (CatBoost (CAT))	0.7936	0.8504	0.8210	0.8083	0.8295
	H2O.ai (Generalized Linear Modeling (GLM))	0.8139	0.8550	0.8339	0.8251	0.8084
	Proposed Model (Group 5-CSS Type 2) (Stacking: 1st layer: LGBM, DT, SGD, Meta: LR)	**0.9330**	**0.9307**	**0.9318**	0.8269	**0.9121**
Average	Logistic Regression (LR)	0.8329	0.7264	0.7758	**0.8014**	0.8148
	XGBoost (XGB)	0.8225	0.8283	0.8252	0.5668	0.8227
	PyCaret	0.8024	0.8324	0.8170	0.6889	0.7971
	H2O.ai	0.8112	0.8458	0.8281	0.6903	0.7775
	Average of 14 Single Classifiers	0.8127	0.7501	0.7705	0.6101	0.7627

Table A2. *Cont.*

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Average	Average of 2 AutoML Approaches	0.8068	0.8391	0.8226	0.6896	0.7873
	Average of All Single Classifiers and AutoML	0.8120	0.7612	0.7770	0.6200	0.7658
	Proposed Model (CSS Type 2)	**0.8717**	**0.8742**	**0.8729**	0.7196	**0.8326**

Table A3. Detailed performance results of Type 3 severity for Groups 1–5.

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Group 1	Logistic Regression (LR)	**0.7172**	0.5643	0.6316	**0.6878**	0.6396
	Support Vector Machine 1 (SVM 1)	0.7100	0.6355	0.6707	0.6171	0.6429
	Support Vector Machine 2 (SVM 2)	0.6593	0.4865	0.5599	0.5579	0.6078
	Stochastic Gradient Descent (SGD)	0.7091	0.5652	0.6290	0.6739	0.6322
	K-Nearest Neighbor (KNN)	0.6665	0.6161	0.6403	0.4775	0.5241
	Multi-Layer Perceptron (MLP)	0.7167	0.5440	0.6185	0.6972	0.6431
	Gaussian Naïve Bayes (GNB)	0.6968	0.6342	0.6640	0.5810	0.6371
	Decision Tree (DT)	0.6792	0.6232	0.6500	0.5162	0.5240
	Extra Trees Classifier (ExTree)	0.6803	0.6152	0.6461	0.5147	0.5269
	XGBoost (XGB)	0.6709	0.7050	0.6875	0.3913	0.6458
	Light Gradient Boosting Machine (LGBM)	0.6825	0.7098	0.6959	0.4239	0.6568
	Random Forest (RF)	0.6787	0.6731	0.6759	0.4709	0.6331
	AdaBoost (ADA)	0.6787	0.6731	0.6759	0.4709	0.5697
	CatBoost (CAT)	0.6767	0.6904	0.6835	0.4419	0.6422
	PyCaret (Logistic Regression (LR))	0.7109	0.7509	0.7304	0.6571	0.6461
	H2O.ai (Stacking: 1st layer: 5 GBM, 1 GLM, 2 DRF, Meta: GLM)	0.6849	0.7907	0.7340	0.4725	0.5425
	Proposed Model (Group 1-CSS Type 3) (Stacking: 1st layer: LGBM, GNB, 2nd layer: MLP, LR, Meta: LR)	0.7157	**0.7691**	**0.7414**	0.5333	**0.6584**
Group 2	Logistic Regression (LR)	0.7299	0.5263	0.6116	**0.7353**	0.6900
	Support Vector Machine 1 (SVM 1)	0.7264	0.5674	0.6371	0.7011	0.6913
	Support Vector Machine 2 (SVM 2)	0.6197	0.6731	0.6453	0.2271	0.6385
	Stochastic Gradient Descent (SGD)	0.7213	0.4644	0.5650	0.7421	0.6825
	K-Nearest Neighbor (KNN)	0.6862	0.6435	0.6642	0.4971	0.5649
	Multi-Layer Perceptron (MLP)	0.7329	0.5254	0.6120	0.7329	0.6859
	Gaussian Naïve Bayes (GNB)	0.7201	0.5798	0.6424	0.6779	0.6700
	Decision Tree (DT)	0.6919	0.6333	0.6613	0.5465	0.5428
	Extra Trees Classifier (ExTree)	0.6618	0.5913	0.6246	0.4834	0.5116
	XGBoost (XGB)	0.6957	0.7253	0.7102	0.4394	0.6972
	Light Gradient Boosting Machine (LGBM)	0.7013	0.7306	0.7157	0.4510	**0.7096**
	Random Forest (RF)	0.6982	0.7284	0.7130	0.4621	0.7042
	AdaBoost (ADA)	0.6982	0.7284	0.7130	0.4621	0.6159
	CatBoost (CAT)	0.6997	0.7191	0.7093	0.4552	0.7089
	PyCaret (Gaussian Naïve Bayes (GNB))	0.7328	0.7717	0.7517	0.6951	0.6897
	H2O.ai (Stacking: 1st layer: 22 GBM, 1 GLM, 2 DRF, Meta: GLM)	0.7001	0.7654	0.7313	0.4936	0.6975
	Proposed Model (Group 2-CSS Type 3) (Stacking: 1st layer: LGBM, RF, ADA, 2nd layer: MLP, LR, Meta: LR)	**0.7420**	**0.7749**	**0.7581**	0.5805	0.6914

Table A3. *Cont.*

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Group 3	Logistic Regression (LR)	0.7386	0.5343	0.6251	**0.7472**	0.7017
	Support Vector Machine 1 (SVM 1)	0.7313	0.5458	0.6318	0.7264	0.7101
	Support Vector Machine 2 (SVM 2)	0.6078	0.6577	0.5639	0.2054	0.6346
	Stochastic Gradient Descent (SGD)	0.7243	0.4617	0.6622	0.7515	0.6781
	K-Nearest Neighbor (KNN)	0.6786	0.6466	0.6228	0.4842	0.5539
	Multi-Layer Perceptron (MLP)	0.7189	0.5493	0.1050	0.6778	0.6886
	Gaussian Naïve Bayes (GNB)	0.7284	0.0566	0.6692	0.8529	0.6083
	Decision Tree (DT)	0.6910	0.6488	0.6410	0.5159	0.5405
	Extra Trees Classifier (ExTree)	0.6827	0.6041	0.7158	0.5336	0.5423
	XGBoost (XGB)	0.7025	0.7297	0.7234	0.4512	0.7024
	Light Gradient Boosting Machine (LGBM)	0.7105	0.7368	0.7225	0.4549	0.7084
	Random Forest (RF)	0.7125	0.7328	0.7225	0.4754	0.7236
	AdaBoost (ADA)	0.7125	0.7328	0.7111	0.4754	0.5969
	CatBoost (CAT)	0.7012	0.7213	0.6201	0.4535	0.7112
	PyCaret (Random Forest (RF))	0.7231	0.7897	0.7549	0.4928	0.7128
	H2O.ai (Generalized Linear Modeling (GLM))	0.6954	0.7825	0.7408	0.7036	0.7367
	Proposed Model (Group 3-CSS Type 3) (Stacking: 1st layer: LGBM, RF, ADA, 2nd layer: MLP, LR, Meta: LR)	**0.7566**	**0.7892**	**0.7726**	0.5906	**0.7355**
Group 4	Logistic Regression (LR)	0.7899	0.6249	0.6978	**0.8429**	0.8122
	Support Vector Machine 1 (SVM 1)	0.7884	0.6134	0.6900	0.8038	0.8186
	Support Vector Machine 2 (SVM 2)	0.5916	0.7368	0.6563	0.1413	0.7181
	Stochastic Gradient Descent (SGD)	0.7766	0.6280	0.6944	0.7861	0.7685
	K-Nearest Neighbor (KNN)	0.7340	0.6966	0.7148	0.5933	0.6222
	Multi-Layer Perceptron (MLP)	0.7882	0.6851	0.7330	0.7816	0.8123
	Gaussian Naïve Bayes (GNB)	0.7872	0.1004	0.1781	0.8111	0.7191
	Decision Tree (DT)	0.7615	0.7174	0.7388	0.6804	0.6283
	Extra Trees Classifier (ExTree)	0.7308	0.6568	0.6918	0.6370	0.5978
	XGBoost (XGB)	0.7915	0.7948	0.7931	0.6450	0.8660
	Light Gradient Boosting Machine (LGBM)	0.7916	0.7917	0.7916	0.6607	0.8744
	Random Forest (RF)	0.7908	0.7890	0.7899	0.6780	0.8541
	AdaBoost (ADA)	0.7908	0.7890	0.7899	0.6780	0.6274
	CatBoost (CAT)	0.7830	0.7855	0.7842	0.6507	0.6274
	PyCaret (Gaussian Naïve Bayes (GNB))	0.7346	0.7775	0.7554	0.7558	0.7305
	H2O.ai (Stacking: 1st layer: 22 GBM, 1 GLM, 2 DRF, Meta: GLM)	0.6965	0.7966	0.7432	0.7214	0.7484
	Proposed Model (Group 4-CSS Type 3) (Stacking: 1st layer: LGBM, LR, 2nd layer: MLP, Meta: LR)	**0.8239**	**0.8631**	**0.8430**	0.7626	**0.8995**
Group 5	Logistic Regression (LR)	0.7887	0.6240	0.6967	**0.8264**	0.8079
	Support Vector Machine 1 (SVM 1)	0.7923	0.5975	0.6812	0.8183	0.8140
	Support Vector Machine 2 (SVM 2)	0.5917	0.7377	0.6567	0.1414	0.6941
	Stochastic Gradient Descent (SGD)	0.7761	0.6117	0.6842	0.7980	0.7692
	K-Nearest Neighbor (KNN)	0.7237	0.6846	0.7036	0.5756	0.6066
	Multi-Layer Perceptron (MLP)	0.7714	0.7129	0.7410	0.7137	0.8088
	Gaussian Naïve Bayes (GNB)	0.7678	0.3883	0.5158	0.8107	0.6786
	Decision Tree (DT)	0.7639	0.7147	0.7385	0.6815	0.6257
	Extra Trees Classifier (ExTree)	0.7110	0.6417	0.6746	0.5775	0.5797
	XGBoost (XGB)	0.7855	0.7956	0.7905	0.6272	0.8658

Table A3. *Cont.*

Group	Model	Precision	Recall	F1 Score	Specificity	AUC
Group 5	Light Gradient Boosting Machine (LGBM)	0.7854	0.7925	0.7889	0.6363	0.8720
	Random Forest (RF)	0.7786	0.7890	0.7838	0.6391	0.8475
	AdaBoost (ADA)	0.7786	0.7890	0.7838	0.6391	0.6228
	CatBoost (CAT)	0.7820	0.7895	0.7857	0.6477	0.8603
	PyCaret (CatBoost (CAT))	0.7231	0.7883	0.7543	0.6782	0.7362
	H2O.ai (Gradient Boosting Machine (GBM))	0.7054	0.7895	0.7451	0.6926	0.7187
	Proposed Model (Group 5-CSS Type 3) (Stacking: 1st layer: LGBM, SVM2, 2nd layer: MLP, LR, Meta: LR)	**0.8564**	**0.8952**	**0.8754**	0.7068	**0.8834**
Average	Logistic Regression (LR)	0.7529	0.5748	0.6526	**0.7679**	0.7303
	XGBoost (XGB)	0.7292	0.7501	0.7409	0.5108	0.7554
	PyCaret	0.7249	0.7756	0.7493	0.6558	0.7031
	H2O.ai	0.6965	0.7849	0.7389	0.6167	0.6888
	Average of 14 Single Classifiers	0.7220	0.6430	0.6694	0.5923	0.6791
	Average of 2 AutoML Approaches	0.7107	0.7803	0.7441	0.6363	0.6959
	Average of All Single Classifiers and AutoML	0.7206	0.6601	0.6787	0.5978	0.6812
	Proposed Model (CSS Type 3)	**0.7789**	**0.8183**	**0.7981**	0.6348	**0.7736**

Appendix B. Selected Features from Feature Selection Algorithm for All Groups

Table A4. Selected features from feature selection algorithm for all groups.

Groups	Feature Selection	Selected Feature (Clinical Variables)
Group 1	None (9)	AGE, SEX, PREG, PREGW, BMI, SBP, DBP, HRI, TEMPI
	Forward Selection (5)	AGE, PREG, BMI, HRI, TEMPI
	Backward Elimination (5)	AGE, PREGW, SBP, HRI, TEMPI
	RFECV (5)	AGE, BMI, SBP, HRI, TEMPI
	Proposed Algorithm (5)	AGE, BMI, SBP, HRI, TEMPI
Group 2	None (21)	AGE, SEX, PREG, PREGW, BMI, SBP, DBP, HRI, TEMPI, FEVER, COUGH, SPUTUM, ST, RNR, MAM, FM, SOB, HEADA, ACC, VN, DIARR
	Forward Selection (6)	AGE, TEMPI, SOB, COUGH, SPUTUM
	Backward Section (18)	AGE, SEX, PREG, PREGW, BMI, SBP, DBP, HRI, TEMPI, FEVER, COUGH, SPUTUM, ST, RNR, MAM, SOB, HEADA, ACC
	RFECV (16)	AGE, SEX, BMI, SBP, DBP, HRI, TEMPI, FEVER, COUGH, SPUTUM, MAM, SOB, HEADA, VN, DIARR
	Proposed Algorithm (13)	AGE, SEX, BMI, SBP, DBP, HRI, TEMPI, FEVER, COUGH, SPUTUM, MAM, SOB, HEADA
Group 3	None (32)	AGE, SEX, PREG, PREGW, BMI, SBP, DBP, HRI, TEMPI, FEVER, COUGH, SPUTUM, ST, RNR, MAM, FM, SOB, HEADA, ACC, VN, DIARR, DM, HTN, HF, CCD, ASTHMA, COPD, CKD, MALIG, CLD, RDAD, DEMEN
	Forward Selection (9)	AGE, TEMPI, FEVER, FM, SOB, ACC, HTN, RDAD, DEMEN
	Backward Elimination (28)	AGE, SEX, PREG, PREGW, SBP, HRI, TEMPI, FEVER, COUGH, SPUTUM, ST, RNR, MAM, FM, SOB, HEADA, ACC, DIARR, DM, HTN, CCD, ASTHMA, COPD, CKD, MALIG, CLD, DEMEN
	RFECV (26)	AGE, SEX, BMI, SBP, DBP, HRI, TEMPI, FEVER, COUGH, SPUTUM, ST, RNR, MAM, FM, SOB, HEADA, ACC, VN, DIARR, DM, HTN, CCD, ASTHMA, MALIG, DEMEN
	Proposed Algorithm (22)	AGE, SEX, SBP, HRI, TEMPI, FEVER, COUGH, SPUTUM, ST, RNR, MAM, FM, SOB, HEADA, ACC, DIARR, DM, HTN, CCD, ASTHMA, MALIG, DEMEN

Table A4. *Cont.*

Groups	Feature Selection	Selected Feature (Clinical Variables)
Group 4	None (26)	AGE, SEX, PREG, PREGW, BMI, SBP, DBP, HRI, TEMPI, FEVER, COUGH, SPUTUM, ST, RNR, MAM, FM, SOB, HEADA, ACC, VN, DIARR, HGB, HCT, LYMPHO, PLT, WBC
	Forward Selection (13)	AGE, SEX, SBP, TEMPI, COUGH, MAM, SOB, ACC, HGB, HCT, LYMPHO, PLT, WBC
	Backward Section (24)	AGE, SEX, PREG, PREGW, BMI, SBP, DBP, FEVER, COUGH, SPUTUM, ST, RNR, MAM, FM, SOB, HEADA, ACC, VN, DIARR, HGB, HCT, LYMPHO, PLT, WBC
	RFECV (15)	AGE, SEX, BMI, SBP, DBP, HRI, TEMPI, FEVER, COUGH, SOB, HGB, HCT, LYMPHO, PLT, WBC
	Proposed Algorithm (16)	AGE, SEX, BMI, SBP, DBP, TEMPI, FEVER, COUGH, MAM, SOB, ACC, HGB, HCT, LYMPHO, PLT, WBC
Group 5	None (37)	AGE, SEX, PREG, PREGW, BMI, SBP, DBP, HRI, TEMPI, FEVER, COUGH, SPUTUM, ST, RNR, MAM, FM, SOB, HEADA, ACC, VN, DIARR, DM, HTN, HF, CCD, ASTHMA, COPD, CKD, MALIG, CLD, RDAD, DEMEN, HGB, HCT, LYMPHO, PLT, WBC
	Forward Selection (13)	AGE, TEMPI, ST, MAM, SOB, ACC, DM, DEMEN, HGB, HCT, LYMPHO, PLT, WBC
	Backward Elimination (32)	AGE, SEX, PREG, PREGW, BMI, SBP, DBP, TEMPI, FEVER, COUGH, SPUTUM, ST, RNR, MAM, FM, SOB, HEADA, ACC, DM, HTN, HF, ASTHMA, COPD, CKD, MALIG, RDAD, DEMEN, HGB, HCT, LYMPHO, PLT, WBC
	RFECV (22)	AGE, SEX, BMI, TEMPI, FEVER, ST, SOB, HEADA, ACC, VN, DIARR, DM, ASTHMA, COPD, MALIG, CLD, DEMEN, HGB, HCT, LYMPHO, PLT, WBC
	Proposed Algorithm (19)	AGE, SEX, BMI, TEMPI, FEVER, ST, MAM, SOB, HEADA, ACC, DM, HTN, COPD, MALIG, DEMEN, HGB, HCT, LYMPHO, PLT, WBC

References

1. Kang, E.; Lee, S.Y.; Jung, H.; Kim, M.S.; Cho, B.; Kim, Y.S. Operating protocols of a community treatment center for isolation of patients with coronavirus disease, South Korea. *J. Emerg. Infect. Dis.* **2020**, *26*, 2329–2337. [CrossRef]
2. Hamidi, Z.; Jabraeili-Siahroud, S.; Taati-Alamdari, Y.; Aghbash, P.S.; Shamekh, A.; Baghi, H.B. A comprehensive review of COVID-19 symptoms and treatments in the context of autoimmune diseases. *Virol. J.* **2023**, *20*, 1. [CrossRef]
3. WHO. Living Guidance for Clinical Management of COVID-19. 23 November 2021. Available online: https://www.who.int/publications/i/item/WHO-2019-nCoV-clinical-2021-2 (accessed on 14 November 2023).
4. Rajaraman, S.; Siegelman, J.; Alderson, P.O.; Folio, L.S.; Folio, L.R.; Antani, S.K. Iteratively pruned deep learning ensembles for COVID-19 detection in chest X-rays. *IEEE Access* **2020**, *8*, 115041–115050. [CrossRef]
5. Yao, H.; Zhang, N.; Zhang, R.; Duan, M.; Xie, T.; Pan, J.; Peng, E.; Huang, J.; Zhang, Y.; Xu, X.; et al. Severity detection for the coronavirus disease 2019 (COVID-19) patients using a machine learning model based on the blood and urine tests. *Front. Cell Dev. Biol.* **2020**, *8*, 683. [CrossRef]
6. Brinati, D.; Campagner, A.; Ferrari, D.; Locatelli, M.; Banfi, G.; Cabitza, F. Detection of COVID-19 infection from routine blood exams with machine learning: A feasibility study. *J. Med. Syst.* **2020**, *44*, 135. [CrossRef] [PubMed]
7. Izquierdo, J.L.; Ancochea, J.; Soriano, J.B.; Group, S.C.-R. Clinical characteristics and prognostic factors for intensive care unit admission of patients with COVID-19: Retrospective study using machine learning and natural language processing. *J. Med. Internet Res.* **2020**, *22*, e21801. [CrossRef] [PubMed]
8. Jovanoski, N.; Chen, X.; Becker, U.; Zalocusky, K.; Chawla, D.; Tsai, L.; Borm, M.; Neighbors, M.; Yau, V. Severity of COVID-19 and adverse long-term outcomes: A retrospective cohort study based on a US electronic health record database. *BMJ Open* **2021**, *11*, e056284. [CrossRef]
9. Aktar, S.; Ahamad, M.M.; Rashed-Al-Mahfuz, M.; Azad, A.; Uddin, S.; Kamal, A.; Alyami, S.A.; Lin, P.I.; Islam, S.M.S.; Quinn, J.M.; et al. Machine learning approach to predicting COVID-19 disease severity based on clinical blood test data: Statistical analysis and model development. *JMIR Med. Inform.* **2021**, *9*, e25884. [CrossRef] [PubMed]
10. Lassau, N.; Ammari, S.; Chouzenoux, E.; Gortais, H.; Herent, P.; Devilder, M.; Soliman, S.; Meyrignac, O.; Talabard, M.P.; Lamarque, J.P.; et al. Integrating deep learning CT-scan model, biological and clinical variables to predict severity of COVID-19 patients. *Nat. Commun.* **2021**, *12*, 634. [CrossRef]

11. Li, K.; Liu, X.; Yip, R.; Yankelevitz, D.F.; Henschke, C.I.; Geng, Y.; Fang, Y.; Li, W.; Pan, C.; Chen, X.; et al. Early prediction of severity in coronavirus disease (COVID-19) using quantitative CT imaging. *Clin. Imaging* **2021**, *78*, 223–229. [CrossRef]
12. Kremer, S.; Lersy, F.; de Sèze, J.; Ferré, J.C.; Maamar, A.; Carsin-Nicol, B.; Collange, O.; Bonneville, F.; Adam, G.; Martin-Blondel, G.; et al. Brain MRI findings in severe COVID-19: A retrospective observational study. *Radiology* **2020**, *297*, 242–251. [CrossRef]
13. An, C.; Oh, H.C.; Chang, J.H.; Oh, S.-J.; Lee, J.M.; Han, C.H.; Kim, S.W. Development and validation of a prognostic model for early triage of patients diagnosed with COVID-19. *Sci. Rep.* **2021**, *11*, 21923. [CrossRef]
14. An, C.; Lim, H.; Kim, D.W.; Chang, J.H.; Choi, Y.J.; Kim, S.W. Machine learning prediction for mortality of patients diagnosed with COVID-19: A nationwide Korean cohort study. *Sci. Rep.* **2020**, *10*, 18716. [CrossRef]
15. Bean, J.; Kuri-Cervantes, L.; Pennella, M.; Betts, M.R.; Meyer, N.J.; Hassan, W.M. Multivariate indicators of disease severity in COVID-19. *Sci. Rep.* **2023**, *13*, 5145. [CrossRef]
16. Tahsin, L.; Roy, S. Prediction of COVID-19 severity level using the XGBoost algorithm: A machine learning approach based on the SIR epidemiological model. In Proceedings of the Intelligent Systems and Sustainable Computing (ICISSC) 2021, Hyderabad, India, 24–25 September 2021; pp. 1–8.
17. Moulaei, K.; Shanbehzadeh, M.; Mohammadi-Taghiabad, Z.; Kazemi-Arpanahi, H. Comparing machine learning algorithms for predicting COVID-19 mortality. *BMC Med. Inform. Decis. Mak.* **2022**, *22*, 2. [CrossRef]
18. Barough, S.S.; Safavi-Naini, S.A.A.; Siavoshi, F.; Tamimi, A.; Ilkhani, S.; Akbari, S.; Ezzati, S.; Hatamabadi, H.; Pourhoseingholi, M.A. Generalizable machine learning approach for COVID-19 mortality risk prediction using on-admission clinical and laboratory features. *Sci. Rep.* **2023**, *13*, 2399. [CrossRef] [PubMed]
19. Banoei, M.M.; Dinparastisaleh, R.; Zadeh, A.V.; Mirsaeidi, M. Machine-learning-based COVID-19 mortality prediction model and identification of patients at low and high risk of dying. *Crit. Care* **2021**, *25*, 328. [CrossRef] [PubMed]
20. Yan, L.; Zhang, H.T.; Goncalves, J.; Xiao, Y.; Wang, M.; Guo, Y.; Sun, C.; Tang, X.; Jing, L.; Zhang, M. An interpretable mortality prediction model for COVID-19 patients. *Nat. Mach. Intell.* **2020**, *2*, 283–288. [CrossRef]
21. Shang, W.; Dong, J.; Ren, Y.; Tian, M.; Li, W.; Hu, J.; Li, Y. The value of clinical parameters in predicting the severity of COVID-19. *J. Med. Virol.* **2020**, *92*, 2188–2192. [CrossRef]
22. Zhang, J.; Wang, X.; Jia, X.; Li, J.; Hu, K.; Chen, G.; Wei, J.; Gong, Z.; Zhou, C.; Yu, H.; et al. Risk factors for disease severity, unimprovement, and mortality in COVID-19 patients in Wuhan, China. *Clin. Microbiol. Infect.* **2020**, *26*, 767–772. [CrossRef]
23. Jee, Y.; Kim, Y.-J.; Oh, J.; Kim, Y.-J.; Ha, E.-H.; Jo, I. A COVID-19 mortality prediction model for Korean patients using the nationwide Korean Disease Control and Prevention Agency database. *Sci. Rep.* **2022**, *12*, 3311. [CrossRef] [PubMed]
24. Liang, W.; Yao, J.; Chen, A.; Lv, Q.; Zanin, M.; Liu, J.; Wong, S.; Li, Y.; Lu, J.; Liang, H.; et al. Early triage of critically ill COVID-19 patients using deep learning. *Nat. Commun.* **2021**, *11*, 3543. [CrossRef]
25. Jin, C.; Chen, W.; Cao, Y.; Xu, Z.; Tan, Z.; Zhang, X.; Deng, L.; Zheng, C.; Zhou, J.; Shi, H.; et al. Development and evaluation of an artificial intelligence system for COVID-19 diagnosis. *Nat. Commun.* **2020**, *11*, 5088. [CrossRef] [PubMed]
26. Xu, Q.; Zhan, X.; Zhou, Z.; Li, Y.; Xie, P.; Zhang, S.; Li, X.; Yu, Y.; Zhou, C.; Zhang, L.; et al. AI-based analysis of CT images for rapid triage of COVID-19 patients. *NPJ Digit. Med.* **2021**, *4*, 75. [CrossRef]
27. Al Rahhal, M.M.; Bazi, Y.; Jomaa, R.M.; AlShibli, A.; Alajlan, N.; Mekhalfi, M.L.; Melgani, F. COVID-19 detection in CT/X-ray imagery using Vision Transformers. *J. Pers. Med.* **2022**, *12*, 310. [CrossRef] [PubMed]
28. Guan, W.-J.; Ni, Z.-Y.; Hu, Y.; Liang, W.-H.; Ou, C.-Q.; He, J.-X.; Liu, L.; Shan, H.; Lei, C.-L.; Hui, D.S. Clinical characteristics of coronavirus disease 2019 in China. *N. Engl. J. Med.* **2020**, *382*, 1708–1720. [CrossRef]
29. Wungu, C.D.K.; Khaerunnisa, S.; Putri, E.A.C.; Hidayati, H.B.; Qurnianingsih, E.; Lukitasari, L.; Humairah, I.; Soetjipto. Meta-analysis of cardiac markers for predictive factors on severity and mortality of COVID-19. *Int. J. Infect. Dis.* **2021**, *105*, 551–559. [CrossRef]
30. Bayat, V.; Phelps, S.; Ryono, R.; Lee, C.; Parekh, H.; Mewton, J.; Sedghi, F.; Etminani, P.; Holodniy, M. A severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) prediction model from standard laboratory tests. *Clin. Infect. Dis.* **2021**, *73*, 2901–2907. [CrossRef]
31. Fan, X.-R.; Zuo, J.; He, W.-T.; Liu, W. Stacking based prediction of COVID-19 Pandemic by integrating infectious disease dynamics model and traditional machine learning. In Proceedings of the 2022 5th International Conference on Big Data and Internet of Things (BDIOT '22), Chongqing, China, 12–14 August 2022; Association for Computing Machinery: New York, NY, USA, 2022; pp. 20–26.
32. Gupta, A.; Jain, V.; Singh, A. Stacking Ensemble-Based Intelligent Machine Learning Model for Predicting Post-COVID-19 Complications. *New Gener. Comput.* **2021**, *40*, 987–1007. [CrossRef]
33. Shakhovska, N.; Yakovyna, V.; Chopyak, V. A new hybrid ensemble machine-learning model for severity risk assessment and post-COVID prediction system. *Math. Biosci. Eng.* **2022**, *19*, 6102–6123. [CrossRef]
34. Rahman, T.; Chowdhury, M.E.; Khandakar, A.; Mahbub, Z.B.; Hossain, M.S.A.; Alhatou, A.; Abdalla, E.; Muthiyal, S.; Islam, K.F.; Kashem, S.B.A.; et al. BIO-CXRNET: A robust multimodal stacking machine learning technique for mortality risk prediction of COVID-19 patients using chest X-ray images and clinical data. *Neural Comput. Appl.* **2023**, *35*, 17461–17483. [CrossRef] [PubMed]
35. De Paiva, B.B.; Pereira, P.D.; de Andrade, C.M.; Gomes, V.M.; Souza-Silva, M.V.; Martins, K.P.; Sales, T.L.; de Carvalho, R.L.; Pires, M.C.; Ramos, L.E.; et al. Potential and limitations of machine meta-learning (ensemble) methods for predicting COVID-19 mortality in a large inhospital Brazilian dataset. *Sci. Rep.* **2023**, *13*, 3463. [CrossRef] [PubMed]

36. Abayomi-Alli, O.O.; Damaševičius, R.; Maskeliūnas, R.; Misra, S. An ensemble learning model for COVID-19 detection from blood test samples. *Sensors* **2022**, *22*, 2224. [CrossRef] [PubMed]
37. Kablan, R.; Miller, H.A.; Suliman, S.; Frieboes, H.B. Evaluation of stacked ensemble model performance to predict clinical outcomes: A COVID-19 study. *Int. J. Med. Inf.* **2023**, *175*, 105090. [CrossRef] [PubMed]
38. Ikemura, K.; Bellin, E.; Yagi, Y.; Billett, H.; Saada, M.; Simone, K.; Stahl, L.; Szymanski, J.; Goldstein, D.Y.; Gil, M.R. Using automated machine learning to predict the mortality of patients with COVID-19: Prediction model development study. *J. Med. Internet Res.* **2021**, *23*, e23458. [CrossRef] [PubMed]
39. De Holanda, W.D.; E Silva, L.C.; de Carvalho César Sobrinho, Á.A. Machine learning models for predicting hospitalization and mortality risks of COVID-19 patients. *Expert Syst. Appl.* **2024**, *240*, 122670. [CrossRef]
40. López-Úbeda, P.; Díaz-Galiano, M.C.; Martín-Noguerol, T.; Luna, A.; Ureña-López, L.A.; Martín-Valdivia, M.T. COVID-19 detection in radiological text reports integrating entity recognition. *Comput. Biol. Med.* **2020**, *127*, 104066. [CrossRef]
41. Mermin-Bunnell, K.; Zhu, Y.; Hornback, A.; Damhorst, G.; Walker, T.; Robichaux, C.; Mathew, L.; Jaquemet, N.; Peters, K.; Johnson, T.M., 2nd; et al. Use of natural language processing of patient-initiated electronic health record messages to identify patients with COVID-19 infection. *JAMA Netw. Open* **2023**, *6*, e2322299. [CrossRef]
42. Guo, M.; Ma, Y.; Eworuke, E.; Khashei, M.; Song, J.; Zhao, Y.; Jin, F. Identifying COVID-19 cases and extracting patient reported symptoms from Reddit using natural language processing. *Sci. Rep.* **2023**, *13*, 13721. [CrossRef]
43. Kim, G.-W.; Lee, D.-H. Personalised health document summarisation exploiting Unified Medical Language System and topic-based clustering for mobile healthcare. *J. Inf. Sci.* **2018**, *44*, 619–643. [CrossRef]
44. Kim, G.-W.; Lee, D.-H. Intelligent health diagnosis technique exploiting automatic ontology generation and web-based personal health record services. *IEEE Access* **2019**, *7*, 9419–9444. [CrossRef]
45. Chen, C.W.; Tsai, Y.H.; Chang, F.R.; Lin, W.C. Ensemble feature selection in medical datasets: Combining filter, wrapper, and embedded feature selection results. *Expert Syst.* **2020**, *37*, e12553. [CrossRef]
46. El-Rashidy, N.; El-Sappagh, S.; Abuhmed, T.; Abdelrazek, S.M.; El-Bakry, H.M. Intensive care unit mortality prediction: An improved patient-specific stacking ensemble model. *IEEE Access* **2020**, *8*, 133541–133564. [CrossRef]
47. Abdellatif, A.; Abdellatef, H.; Kanesan, J.; Chow, C.-O.; Chuah, J.H.; Gheni, H.M. An effective heart disease detection and severity level classification model using machine learning and hyperparameter optimization methods. *IEEE Access* **2022**, *10*, 79974–79985. [CrossRef]
48. Khushi, M.; Shaukat, K.; Alam, T.M.; Hameed, I.A.; Uddin, S.; Luo, S.; Yang, X.; Reyes, M.C. A comparative performance analysis of data resampling methods on imbalance medical data. *IEEE Access* **2021**, *9*, 109960–109975. [CrossRef]
49. Heo, J.; Han, D.; Kim, H.-J.; Kim, D.; Lee, Y.-K.; Lim, D.; Hong, S.O.; Park, M.-J.; Ha, B.; Seog, W. Prediction of patients requiring intensive care for COVID-19: Development and validation of an integer-based score using data from Centers for Disease Control and Prevention of South Korea. *J. Intensive Care* **2021**, *9*, 16. [CrossRef]
50. Kim, J.; Lim, H.; Ahn, J.-H.; Lee, K.H.; Lee, K.S.; Koo, K.C. Optimal triage for COVID-19 patients under limited health care resources with a parsimonious machine learning prediction model and threshold optimization using discrete-event simulation: Development study. *JMIR Med. Inform.* **2021**, *9*, e32726. [CrossRef]
51. Ali, M. PyCaret: An Open Source, Low-Code Machine Learning Library in Python. 2020. Available online: https://pycaret.readthedocs.io/en/latest/index.html (accessed on 18 February 2024).
52. LeDell, E.; Poirier, S. H_2O AutoML: Scalable Automatic Machine Learning. In Proceedings of the AutoML Workshop at ICML, Vienna, Austria, 17–18 July 2020; Volume 2020.