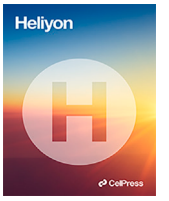
Heliyon 9 (2023) e18402

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon



Research article

Medical emergency department triage data processing using a machine-learning solution

Andreea Vântu [a], Anca Vasilescu [b,*], Alexandra Băicoianu [b]

[a] Faculty of Mathematics and Computer Science, Transilvania University of Braşov, Romania
[b] Department of Mathematics and Computer Science, Transilvania University of Braşov, Romania

ARTICLE INFO

Keywords:
Emergency medicine
Triage
Clinical decision support
Patient medical record
Medical data processing
Machine learning
Supervised learning algorithms

ABSTRACT

Over the years, artificial intelligence has demonstrated its ability to overcome many challenges in our day-to-day life. The evolution of it inquired more studies about Machine Learning possible solutions for different domains, including health care. The increasing demand for artificial intelligence solutions has brought accessibility to loads of data, including clinical data. The availability of medical records facilitates new opportunities to explore Machine Learning models and their abilities to process a significant amount of data and to identify patterns with the purpose of solving a medical problem. Understanding the applicability of artificial intelligence on this type of data has to be a compelling aim for emergency medicine clinicians. This paper focuses on the general clinical problem of the complex correlation between medical records and later diagnosis and, especially, on the process of emergency department triage which uses the Emergency Severity Index (ESI) as triage protocol. This study presents a comparison between three different Machine Learning models, such as Logistic Regression, Random Forest Tree and NN-Sequentail, with the purpose of classifying patients with an emergency code. We conducted four experiments because of imbalanced data. A web-based application was developed to improve the triage process after our theoretical and exploratory results. Overall, in all experiments, the NN-Sequential model had better results, having, in the first experiment, a ROC-AUC score for each ESI emergency code of: 0.59%, 0.76%, 0.71%, 0.78% 0.64%. After applying methods to balance the data, the model yielded a ROC-AUC score for each emergency code of 0.72%, 0.75%, 0.69%, 0.74%, 0.78%. In the last experiment consisting of a three-class classification problem, the NN-Sequential and Random Forest Tree models had similar metric outcomes, and the NN-Sequential algorithm had a ROC-AUC score for each emergency code of: 0.76%, 0.72%, 0.84%. Without any doubt, our research results presented in this paper endorse this tremendous curiosity in Machine Learning applications to enrich aspects of emergency medical care by applying specific methods for processing both medical data and medical records.

1. Introduction

From a theoretical point of view, Machine Learning provides various learning algorithms for different types of problems [1], [2], but practical solutions emerge to increase Machine Learning involvement and impact on quality of life. Artificial Intelligence

* Corresponding author.
E-mail addresses: andreea.vantu@unitbv.ro (A. Vântu), avasilescu@unitbv.ro (A. Vasilescu), a.baicoianu@unitbv.ro (A. Băicoianu).

https://doi.org/10.1016/j.heliyon.2023.e18402
Received 11 January 2023; Received in revised form 17 July 2023; Accepted 17 July 2023

Available online 22 July 2023

has many real-life applications, from solutions for renewable energy system [3] to Emergency Department improvements [4] [5] [6]. Over the last two decades, many research papers about the applications of Machine Learning in medicine and health care were published; for example, from [7] in 2001 to [8] in 2022, specific medical data processing methods were revealed. Part of them focuses on medically diagnosing a patient with a specific disease, like predicting diabetes [9] or trying to improve the Emergency Department flow with technology [10]. It is important to accumulate and recall the individuals' medical records efficiently and, therefore, accurately evaluate and classify symptoms in emergency medicine and services, which can be consistently improved by implementing Machine Learning algorithms. In papers like [10] or [11], predicting the admission of a patient after the Emergency Department triage and also predicting the emergency levels using the Machine Learning approaches are covered in various contexts. The admission of a patient represents a binary problem with a simple output as *Admit* or *Discharge*.

This study aims to research supervised Machine Learning models applied to a multi-class classification problem, such as predicting an emergency code to one individual in the specific context of the Romanian emergency medical system. This research focuses on a Machine Learning approach to the Emergency Room (ER) or an Emergency Department (ED) triage and follows the ideas from [12] and [13] that refer to a software web-based application developed to enhance the emergency triage process. For that particular work, information and data were collected from the local pediatric hospital, along with the demands and needs addressed by the hospital management team. The project was developed to digitize the workflow. That software covered three main aspects: (1) showing a waiting list according to every patient's assigned emergency code, (2) after having symptoms as input, the application made an emergency code suggestion, and (3) creating the patient file that could run on smart devices.

The demand for Machine Learning algorithms applied to emergency data has increased over the years [4]. However, there are still gaps in the literature, and more studies need to be conducted on this subject. By presenting this study, we want to evaluate Machine Learning models on classifying patients with an emergency severity index [14]. The proposed solution is based on medical records processing in order to classify patients on one of the 5-level ESI emergency codes [15] using supervised learning algorithms and also having a basic web interface for the clinicians as end-users to early monitor those patients. The better an AI-based prediction application is, the more convincing it could be, and eventually, clinicians could rely on the accuracy of a Machine Learning algorithm [16]. It yields that the research has to provide materials and methods, models and metrics, and computational experiments, which are significant from clinicians' perspectives. The next sections of our paper successively follow these topics. Three classifying supervised Machine Learning models and also four dataset approaches are chosen, and appropriate experiments are developed in order to provide patient classification emergency code as accurately as possible.

2. Materials and methods

The solution proposed by this research is meant to solve a multi-class classification problem where the dependent variable consists of 5 classes identified by numbers from 1 to 5. All necessary processes and the implementation were done using Python programming language (Python 3.10.6 version) and its corresponding packages [17].

2.1. Problem analysis

The Emergency Department is one of the most important decision points in a health care system. It is developed using a triage algorithm which prioritizes emergencies. This triage algorithm differs from country to country. In Romania, the triage protocol is based on ESI algorithm. ESI stood for *Emergency Severity Index* and was initially developed in 1999 [15]. It consists of 5-level emergency codes. *Level 1*, the red one, is the most critical level, and the intervention should occur immediately because the patient is in a life-or-death situation. The second level, or the yellow one, means that the patient has a high risk of deterioration, but the patient can wait up to 10 minutes. *Level 3*, the green level, means the patient needs urgent care, but they can wait up to 30 minutes without risking their lives. *Level 4*, or the blue one, represents a patient with a stable health status who requires one resource, and it is considered a non-urgent situation so that the patient can wait up to one hour. The last level, the white one, is also considered non-urgent, and here the patient can wait even two hours. Each patient follows the triage protocol represented here in Fig. 1, and the triage clinician decides to assign an emergency code out of all 5 codes based on specific rules (chief complaints, age, past medical history, triage vital signs, present treatment etc.) following the triage questions.

Overcrowding is the main issue of the Emergency Department. The ESI triage protocol is meant to be as accurate as possible, but it is prone to human error, especially when the medical staff is facing a large number of people at a time. Due to these issues, the health status of a patient can be easily overestimated or underestimated. Overestimation means using more medical resources, but underestimation could result in negative outcomes such as deaths [19].

2.2. Data preparation and preprocessing

The dataset used for this research [20] included adult ED visits within two years, specifically March 2014 and July 2017. The data were collected from three emergency departments. The initial paper [21] used this dataset to predict the admission outcome of a patient, but the features are suited to study emergency levels classification.

This dataset contains 972 features and 560486 entries. All information was gathered in a 1.32 GB CSV file. All these features can be grouped, and they determine separate bigger categories such as demographics, past medical history, historical vitals, triage vital signs, outpatient medication, chief complaints, historical labs, imaging, and hospital usage.

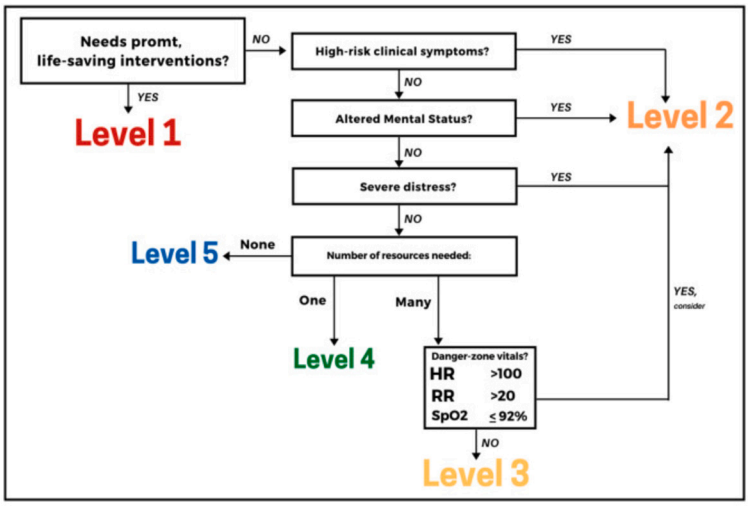


Fig. 1. ESI triage protocol [18].

Table 1
Data analysis.

Category	Attribute	Data type
Response variable	esi	Categorical
Past medical history	From 2sndarymalig to whtblooddx	Categorical
Demographics	age	Numeric
	gender	Categorical
	arrivalmode	Categorical
	previousdispo	Categorical
Triage Vital Signs	triage_vital_hr	Numeric
	triage_vital_rr	Numeric
	triage_vital_sbp	Numeric
	triage_vital_dbp	Numeric
	triage_vital_o2	Numeric
	triage_vital_temp	Numeric
Chief Complaints	From cc_abdominalcramping to cc_wristpain	Categorical

This study does not include all the information provided by this dataset. We have structured and removed the irrelevant features, and we only kept these categories: demographics, past medical history, triage vital signs and chief complaints. This selection is represented in Table 1, keeping those attributes which are more suitable for the Romanian ER patient file, with the intention of further work to fitting that medical system care. Therefore, only 409 features were selected for this research, and all the entries were kept. The resulting dataset is a mix of numerical, binary and text values. For a good workflow, the data types were determined.

The first step in this problem was to understand the variables, assess them and do some basic exploratory data analysis. The starting point consists of plotting the correlation matrix of the trained dataset and using the correlation coefficient's warnings and implications. The response variable is also called the dependent variable because the outcome of it depends on the other attributes. In this case, the *esi* response depends on the symptoms of the patient, for example.

One can evaluate the relationship between each target label and different features using correlations and selecting those features that have the strongest values. For this study, we started analyzing the relationships with triage vital signs, past medical history and chief complaints. The corresponding correlation coefficients with triage vital signs can be seen in Fig. 2.

Next, we scanned the results while looking at past medical history and chief complaints. Because there were plenty of features in the dataset, we selected only ten features for each category (see Fig. 3 and Fig. 4).

Chief complaints represent the symptoms which patients describe when they arrive at the emergency room or department. Back pain, rash and sore throat are the closest ones to 1. The very purpose of the data preprocessing phase is to turn the raw data into a more understandable, useful and efficient format for ML models. Such as, after all the necessary information regarding the classification of one of the 5 emergency codes was gathered from the original data frame, the dataset was split into training and test set. This step was accomplished using *sklearn* library within Python. The training set represents 80% of the data, and the rest of 20% represents the test set. Both training and test sets were preprocessed. The preprocessing step included: handling missing values, scaling the data, and handling text attributes.

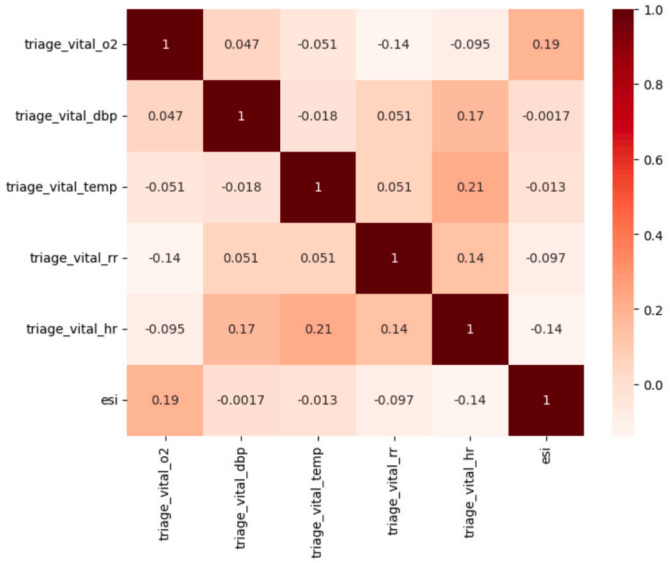


Fig. 2. ESI - Vital Triage Signs Correlation Matrix.

Firstly, it is important to see if the dataset has any missing values and the amount of them because some models do not know how to handle them, and the performance of the algorithms could be affected accordingly. There are many options to treat this case, as explained in [22].

The dataset in this study had missing values for the following features: *esi*, *age*, *arrivalmode*, *vital_triage_hr*, *vital_triage_rr*, *vital_triage_sbp*, *vital_triage_dbp*, *vital_triage_o2*, *vital_triage_temp*, plus some chief complaints attributes. For the *esi* attribute, we decided to drop all the entries with missing values because imputing the NaN value for this *Categorical* variable to the most frequent or median, does not necessarily represent the actual truth which would lead to a false reality. The records which had missing values on *esi* level had missing values on the other columns listed above. Hence, dropping those records fixed the NaN problem on the other features. The remaining missing values on the *age* column were treated in the same way because there were a few records, and it would not affect the outcome of the model's prediction in a significant way. The missing values presented by the triage vital signs were imputed with the median using the class *SimpleImputer* from Python *sklearn.impute* package. We chose median because these are continuous numeric attributes.

Then all the *Numeric* attributes were scaled using the *sklearn* object *StandardScaler()*. Scaling the data is not mandatory, but having big differences between the values could cost the performance of the model. Many Machine Learning models would have a better performance if scaling numerical input has been done before [23]. The dataset contains 7 *Numeric* features with values starting from 18 and going up to 105, and all the other features are *Categorical* having binary values such as 0, 1. We tested the given dataset with the feature selection, and we imputed the missing values, but the data was not scaled, so the model started to perform poorly. Therefore, in order to have a good or decent model performance, the scale was necessary.

Our dataset has three columns with *Categorical* values of type *String*. These columns are: *gender*, *arrivalmode* and *previousdispo*. We used the *OneHotEncoder* class to handle these categorical text features transforming them into *Numeric* ones. It will derive the category based on the unique values in each feature. For example, *gender* feature has only two possible and unique values: male and female. Using *OneHotEncoder* this column will be transformed into two columns with the names: *gender_male* and *gender_female*. These two new columns will be filled with 0 and 1 according to the values on the original column. The same will happen with the other text columns, each unique value being a new feature in the dataset.

The missing values within *arrivalmode* were treated with *SimpleImputer* class but with the most frequent strategy that can be used with both string and numeric types. The median strategy works only with numeric attributes.

3. Models development and evaluation

Since this research project focuses on a multi-class classification problem, namely predicting the patient's emergency code, it has been important to choose the model that suits the problem and has a good predicting score. Until then, one has to test different algorithms to see which one has a better performance. Supervised learning represents those types of Machine Learning in which machines are trained using well-labelled training data. This type of learning implies already-known output data.

For this study, we chose three classifying supervised Machine Learning models, particularly: Neural Network Models (i.e. NN Sequential Model), Decision Trees (i.e. Random Forest Tree Algorithm) and Regression Algorithms (i.e. Logistic Regression Algorithm).

Neural Networks models are designed and named after human biological neurons. They work using a defined number of layers, and each layer contains a precise number of neurons. They are applied to numerous real-world problems, from classifying to facial recognition. Random Forest Tree models represent groups of many individual decision trees that work together to predict the outcome

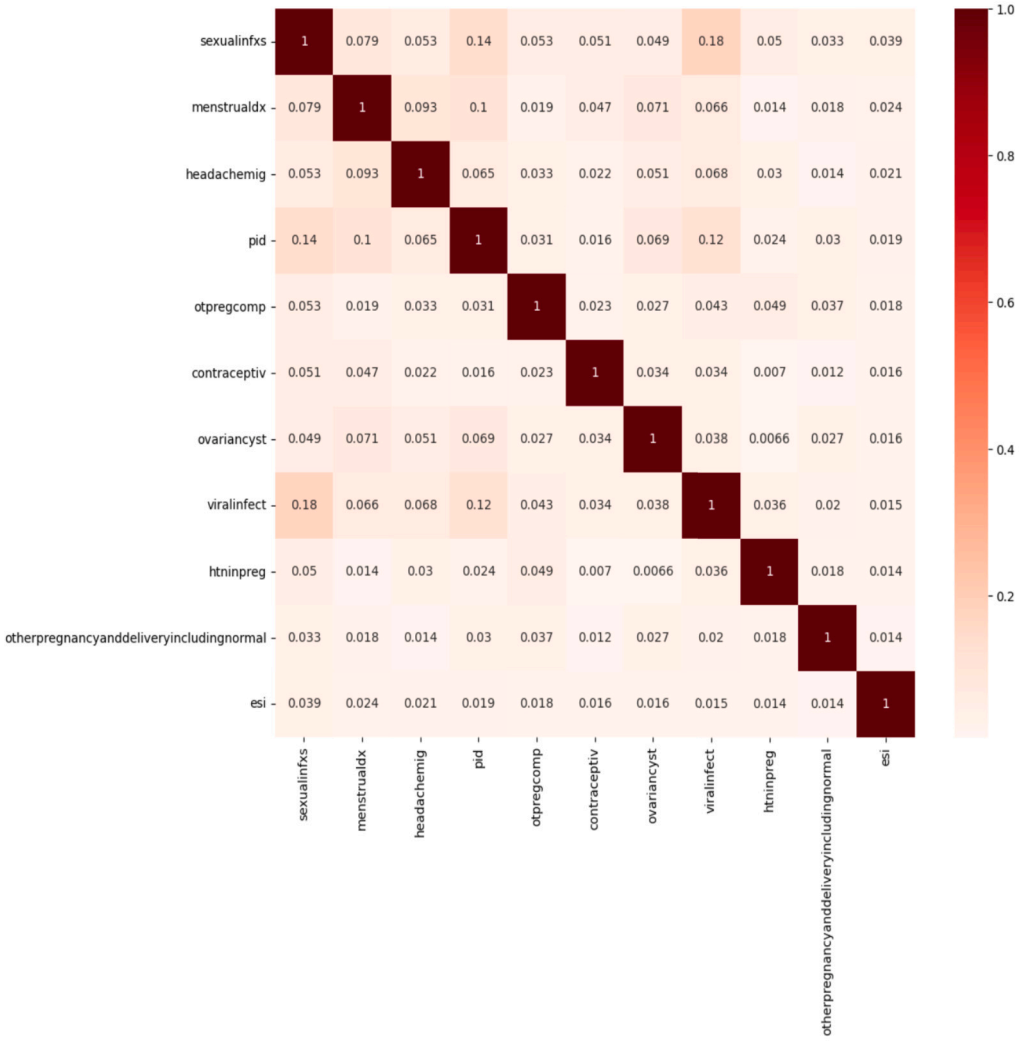


Fig. 3. ESI - Past Medical History Correlation Matrix.

of the input. The logic behind them is simple as it consists in evaluating nodes of a tree. Regression algorithms are used to assess the relationship between features and dependent variables, and they have applicability in fields such as forecasting, trend analysis and so on. We chose Logistic Regression for this study because it is a regression algorithm that can be used for classifying problems.

Usually, Logistic Regression is good for the first guess, it is a simple way to give some results, and it is a popular approach among medical data processing [24] [25] [26]. In addition, Random Forest Tree is a cheap variant, they tend to learn too well, so you usually need a comparative model to contrast them. In particular, one of the advantages offered by a decision tree algorithm is a lower prediction error due to its ability to recognise effects between a predictor and target variable [27]. The NN-Sequential model is sufficient for exploring the given data.

Once the models are chosen, one should take care of the measurements. When we think about a Machine Learning model and its prediction performance, most of the time, we think about accuracy. *Accuracy* is not necessarily the ultimate metric which tells if the model is good or bad, it is important to explore other metrics to give conclusions. A model could give amazing results evaluating one metric such as the accuracy score, but it could act poorly when it comes to another measurement like *F1-score*, which could give relevant insights about the dataset and about the problem in the first place.

This study, as we previously mentioned, treats a multi-class classification problem so that, for each model, the ROC (*Receiver Operating Characteristic*)-AUC (*Area Under the ROC curve*) score, *precision*, *recall*, confusion matrix and *specificity* were evaluated. These metrics prove to be suitable for medical data showing relevant results towards this direction, and besides, they provide clinicians with compelling insight. Some studies only measure the ROC-AUC score, but working with medical data should imply more metrics than this one [28]. The ROC-AUC score is a performance measurement telling how much a model is capable of distinguishing the classes. To make an analogy with medical data, a high score after evaluating this metric means the model is well performing at distinguishing patients who have a disease from the ones who are healthy, for instance, or distinguishing patients who were admitted to the hospital from patients who were not [21].

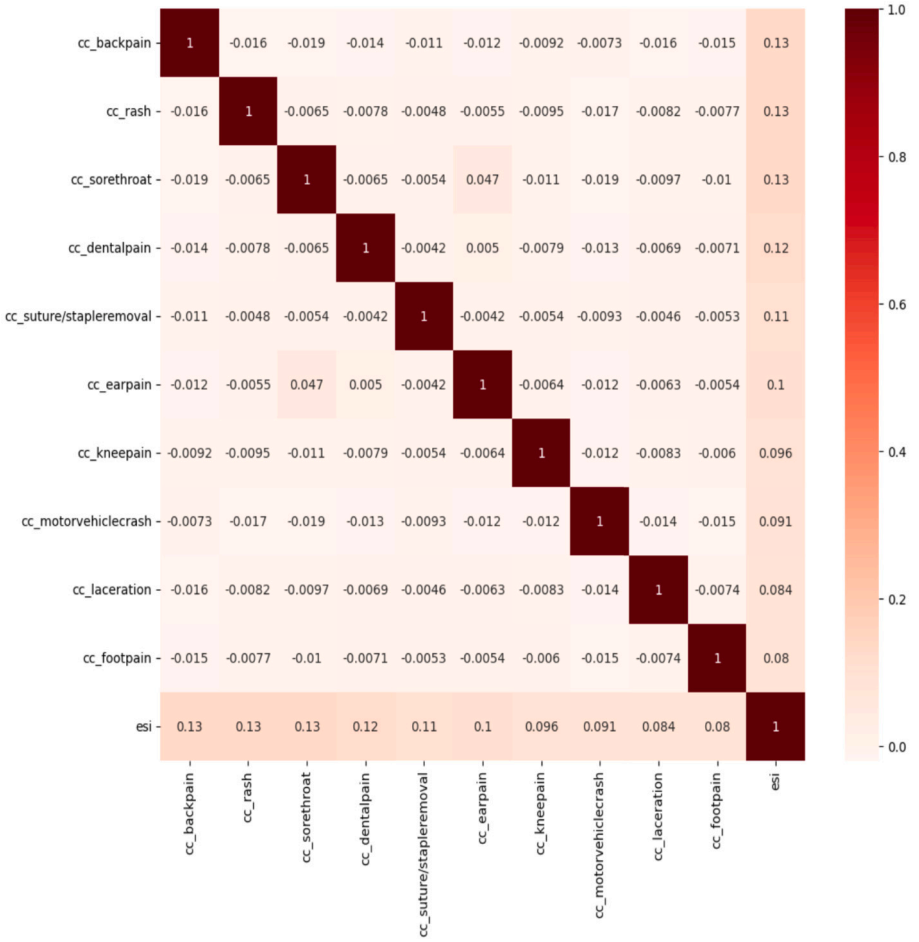


Fig. 4. ESI - Chief Complaints Correlation Matrix.

Considering the usual definitions for the outcomes TP as *True Positive*, FP as *False Positive*, TN as *True Negative*, and FN as *False Negative* and respectively *True Positive Rate* and *False Positive Rate* as in equation (1).

$$TPR = \frac{TP}{TP + FN} \text{ and } FPR = \frac{FP}{FP + FN} \tag{1}$$

the ROC curve maps TPR versus FPR for different classification thresholds, and AUC evaluates the area under the ROC curve between the thresholds corresponding points (0,0) and (1,1) [1]. If AUC yields a high score, then the model performs well at predicting false as false and true as true. An AUC score close to 0 means a poor performance by the model.

All of the metrics offer important details about the performance of the model. For a binary classification problem, it would be easier to pay more attention to a certain metric like *recall*. The admission of a patient could be a binary classification problem if the model has to predict whether the patient was admitted or discharged. *Recall* would show which patients were discharged when they actually needed admission. In our case, the question was not that simple, so we were interested to see all the chosen metrics. It was our target to see how many patients were predicted to have a *Level 3* emergency code, how many were classified on a higher level and also what amount of patients was underestimated, for example. Nevertheless, the ROC-AUC score gives an overview of how accurate the model would be in the future [29]. From a medical perspective, *precision*, *recall* and *specificity* metrics explain more than the performance of the model. All of these three metrics can be calculated using the confusion matrix. A confusion matrix consists of rows which represent the predicted classes and columns which are the true classes. Then, each cell represents the predicted output for each class. Table 2 represents a confusion matrix of our approach.

Considering *Level 2* as an example, the TP, FP, TN, and FN can be applied in a multi-class classification as follows. TP represents the cases predicted as *Level 2*, and the actual output was *Level 2*; its corresponding value following the confusion matrix is TP = 5. FP represents the cases predicted as *Level 2*, and the actual output represents other levels; its corresponding value is FP = 8 + 7 + 3 + 0 = 18. TN represents the cases predicted not as *Level 2* and the actual output was not *Level 2*; its corresponding value is TN = 9 + 3 + 2 + 4 + 8 + 1 + 2 + 4 + 10 + 15 + 1 + 2 + 3 + 4 = 68. FN represents the cases predicted not as *Level 2*, and the actual output was *Level 2*; its corresponding value is FN = 6 + 4 + 3 + 1 = 14.

Table 2
Confusion matrix.

	Level 1	Level 2	Level 3	Level 4	Level 5
Level 1	15	6	3	4	0
Level 2	0	5	8	7	3
Level 3	1	4	9	3	2
Level 4	0	3	4	8	1
Level 5	2	1	2	4	10

Precision and *recall* metrics are also useful when it comes to evaluating the performance of a model in a medical scenario [30] [16]. We understand *precision* as the capability of an algorithm to identify only samples of interest and *recall* as the ability of a classification model to identify all relevant examples. The corresponding evaluations are given by the formulas from equation (2).

$$Precision = \frac{TP}{TP + FP} \text{ and } Recall = \frac{TP}{TP + FN} \tag{2}$$

A model that has a very high *recall* has very less *precision*.

In other words, *recall* describes our TPs from the predicted results, whereas *precision* indicates our TPs from the actual results. Also, *recall* is the equivalent of the TPR parameter of the ROC curve. If we are interested to see the *recall* score, we should ask the right question: out of everything predicted, what is the ratio of missing *Level 2* patients? Same with *precision*: out of all *Level 2* patients, we predicted what is the ratio of *Level 2* patients when they did not represent a *Level 2* risk. A low *precision* means overuse of resources and also overcrowding. A low *recall* means that a patient should have been classified with a *Level 2* emergency code, but another code was assigned to them. Now, overestimating the health status does not necessarily imply a tremendous negative outcome, but again it will lead to overusing resources. Instead, if the patient's health condition is underestimated and a lower code is assigned, this could increase the mortality of the patient. In addition, *precision* and *recall* are better for highly imbalanced datasets [31].

In the medical world, any test for diagnosing people with a disease or not should recognize *sensitivity* and *specificity* metrics. *Sensitivity* measures how often people who have the disease get the result positive, and *specificity* measures the ability to detect negative results for cases which indeed have negative results. We know how bad the first scenario is, with no need for further explanations. In the second case, people who do not have the disease will undergo unnecessary testing or even treatments [32] [28]. In binary classification, *recall* is called *sensitivity*.

Specificity is another metric which can be calculated using the confusion matrix values, by the formula from equation (3).

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

4. Computational experiments and results

For this study, we used an NN-Sequential Model, a Random Forest Tree and a Logistic Regression Algorithm to test the results and choose the one with good performance from the medical point of view. We have evaluated the performance of the models on the given set of data in three ways: using the imbalanced data, handling imbalanced and reducing the volume of the original data. For handling imbalanced data, we did two experiments using two different sampling techniques: SMOTE and ADASYN.

Each Machine Learning model uses different hyper-parameters tuned after an empirical analysis. We conducted many experiments by tuning hyper-parameters, following the concept of trial and error. The final hyperparameters are presented in Table 3. The Logistic Regression algorithm required a 'multinomial' option and 'lbfgs' solver due to the multi-class classification problem. For the NN-Sequential model, we used 'categorical_crossentropy' as a loss function. For the optimizer parameter, we chose an SGD object initialized with the following values: lr = 0.01, decay = 1e-6, momentum = 0.9, nesterov = True. The model was trained on 100 epochs and used a batch size of 128 together with a callback parameter that monitored the accuracy. The network had 4 hidden layers, one input layer and one output layer. The first fifth layers had a *relu* activation function and 30 nodes, and the output layer used a *softmax* activation function and 5 nodes. The input layer received the features' number. For the last algorithm, we chose 50 estimators, meaning there would be 50 decision trees.

In terms of time and space complexity, the Logistic Regression model has a train time complexity of $O(n * m)$, the test time complexity is $O(m)$, and the space complexity of the algorithm is $O(m)$, where n is the number of training data and m is the number of features from the considered dataset.

For the Random Forest Tree approach, we have a training time complexity of $O(k * n * \log n * m)$, the test time complexity is $O(m * k)$, and space complexity is $O(k * depthOfTree)$, where n is the number of training examples, m is the number of features from the considered dataset, and k is the number of decision trees. Random Forest is comparatively faster than other algorithms.

For NN Sequential model, we used the in-build *Sequential() API* from *Keras* library. This is a feed-forward neural network that transfers the data from one layer to another until all sequential steps are finished. The space memory should be equal to the sum of all weights created by the network and batch normalization parameters or other specified hyper-parameters. The total number of parameters used by a neural network could be easily found using the function *summary()*, as we did. Overall, the neural network used in this paper has 12780 parameters after the input layer, the next fourth hidden layers have 930 parameters, and the output layer has 155 parameters. As long as the algorithm is coming from standard libraries, the time complexity of it is difficult to calculate, and, moreover, the result could be inaccurate.

Table 3
Models hyper-parameters.

Model	Hyper-parameter	Value
Logistic Regression	multi_class	multinomial
	solver	lbfgs
	C	10
NN-Sequential	loss	categorical_crossentropy
	optimizer	SGD
	metrics	accuracy
	epochs	100
	batch_size	128
	callback	EarlyStopping
Random Fores Tree	n_estimators	50

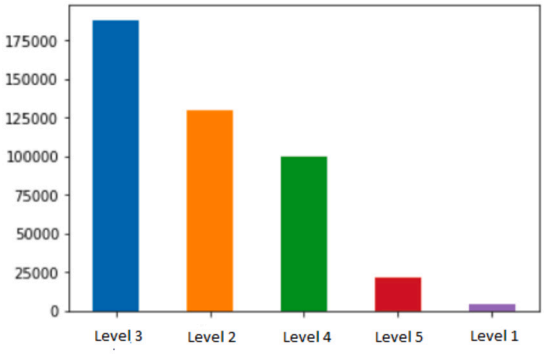


Fig. 5. The distribution of target classes.

The software and libraries used for the following experiments are Python 3.10.6, Jupyter 1.0.0, pandas 2.0.0, numpy 1.23.5, sklearn 0.0.post1, keras 2.12.0, imblearn 0.0.

4.1. Imbalanced data

The distribution of examples across the classes reveals the skewed levels, and they are represented in Fig. 5. Imbalanced data may affect the performance of a Machine Learning model, making a poor prediction. Imbalanced data represent a challenge, especially when it comes to a multi-class classification problem, because the model is prone to perform poorly on the majority class, trying to have better results on minority class [33].

The majority class of our model is represented by *Level 3* emergency code, and the values are decreasing for the other classes. The large amount of examples for *Level 3* is normal because in real life, statistically, *Level 3* is the most frequent emergency code. Here are the numbers for each of the classes: (1.0, 4265), (2.0, 130121), (3.0, 188245), (4.0, 99525), (5.0, 22197). The number of *Level 3* is huge compared to the number of examples of *Level 1*. *Level 2* is closer to *Level 3*, and, sometimes, a given object may be a borderline between these two classes.

Further, the performance of the models is explained starting with the imbalanced dataset and maintaining the same 80%-20% ratio for the train-test split during the experimental evaluations. Our *Experiment 1* is the Neural Network approach that uses a Sequential model with 6 layers provided by *Keras* library. Each of them used, as described at the beginning of Section 4, a *Relu* activation function except the last layer, which used *SoftMax* as an activation function because the target variable has more than one class. Python *SGD* class was used as an optimizer, and the metric was *accuracy*. We trained the model using 100 epochs but also used a callback function to stop the training when the accuracy was dropping. The model stopped training after 41 epochs, having a loss of 0.76 and an accuracy of 0.67.

Fig. 6 displays the confusion matrix after the model was trained and the test set was predicted. The specific results for the *Experiment 1* are outlined in Table 4. *Level 1* has a low *recall* (0.18), which means the FN rate is high. A low *recall* means that 794 patients who needed to be assigned to a *Level 1* emergency code were classified on the lower levels, and only 183 were classified correctly. On the other hand, the *precision* is high (0.71). If the *precision* is high, the FP rate is lower. Hence, out of all *Level 1* predictions, 0.71 were indeed *Level 1*. The *specificity* for *Level 1* is almost 0.99. Therefore, the patients who did not need to be assigned on *Level 1* were correctly assigned to other levels. *Level 2, 3* and *4* have the best scores for *precision* and *recall*.

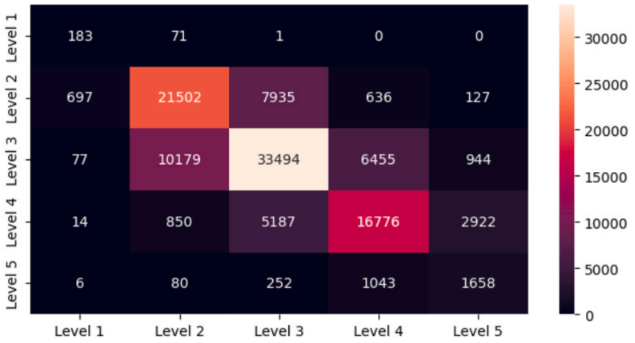


Fig. 6. NN Sequential Model - Confusion Matrix.

Table 4
NN Sequential Model - measurements per classes.

Classes	Recall	Precision	Specificity	ROC-AUC
Level 1	0.18	0.71	0.99	0.59
Level 2	0.65	0.69	0.92	0.76
Level 3	0.71	0.65	0.72	0.71
Level 4	0.67	0.65	0.89	0.78
Level 5	0.29	0.54	0.97	0.64

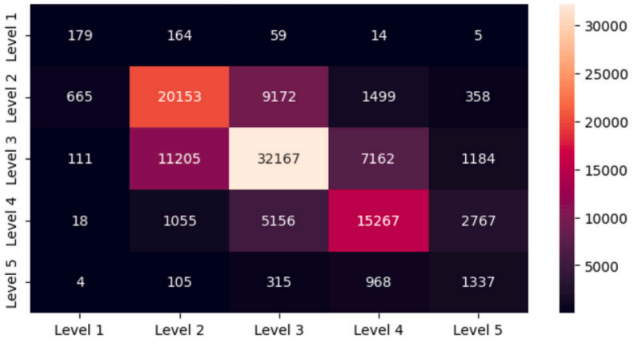


Fig. 7. RFT - Confusion Matrix.

Table 5
RFT - measurements per classes.

Classes	Recall	Precision	Specificity	ROC-AUC
Level 1	0.18	0.42	0.99	0.59
Level 2	0.61	0.63	0.85	0.73
Level 3	0.68	0.62	0.69	0.69
Level 4	0.61	0.62	0.89	0.75
Level 5	0.23	0.48	0.98	0.61

Although *Level 3* has good *precision* and *recall* scores, the *specificity* is not that high. The algorithm is best at distinguishing emergency codes of type 4 with a score of 0.78. *Level 2, 3*, and *4* again have the best scores overall. The confusion matrix shows that *Level 2* and *Level 3* are easily confused.

For our *Experiment 2* involving the Random Forest Tree classifier, we have used the Python *GridSearchCV* function to find the best estimators, and the result was to use 50 estimators. An estimator is a decision tree in the forest. The confusion matrix after training this Random Forest Tree model is presented in Fig. 7.

The specific values from Table 5 yield that this model is not performing as well as the previous one. The *recall* for *Level 1* is still 0.18, but the *precision* is 0.42. There are lower *precision* and *recall* scores for each class compared with the Neural Network measurements.

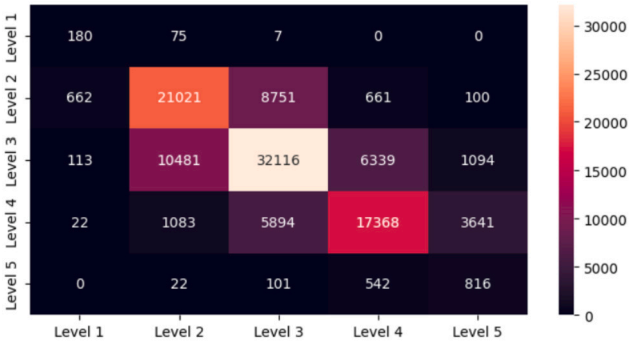


Fig. 8. LR - Confusion Matrix.

Table 6
LR - measurements per classes.

Classes	Recall	Precision	Specificity	ROC-AUC
Level 1	0.18	0.68	0.99	0.59
Level 2	0.64	0.67	0.87	0.75
Level 3	0.68	0.64	0.78	0.70
Level 4	0.69	0.62	0.89	0.78
Level 5	0.14	0.55	0.99	0.56

Looking only at the *specificity*, one would say that this model is very good, but in fact, it is misleading. The ROC-AUC curve is also lower compared to the first measurements. For *Level 3*, the *specificity* is lower, so the model falsely assigns patients on *Level 3*. Looking back to Table 4, we conclude that NN Sequential Model had better scores.

Our *Experiment 3* represents the last model we have considered here. It uses the Logistic Regression algorithm for which we set the parameter *multi_class* to *multinomial* because of the nature of the problem, and we have used *lbfgs* as a solver that handles *L2* or no penalty. *L2* represents regularization added to the algorithm to prevent overfitting. The corresponding confusion matrix is presented in Fig. 8 and we can notice in Table 6 the performance of the model.

Out of all three models, the *Level 4* class has the lowest *recall* score using this particular model, whereas the others are slightly changed. The ROC-AUC curve is similar to the one produced by the Random Forest Tree classifier, and the *precision* score is one percent less than the one produced by the Neural Network. The overall average per measurement is slightly behind the overall measurements from the first model except the *specificity* with an average of 0.9. The NN model had the best results.

The *recall* score for the emergency code of type *Level 1* is always the same (0.18). All three classifiers failed in assigning this type of emergency code to a large number of patients who really needed it. *Level 1* class represents the minority class therefore the low *recall*. Both this metric and *precision* are better to be studied for highly imbalanced datasets [31].

4.2. SMOTE for imbalanced data

Imbalanced data is a great challenge for building a model with considerable performance. The distribution of classes is important, and the difference between them affects the outcome of the algorithm. Following the data provided in the previous section in Fig. 5, the target classes are distributed as percentages in this way: 1% for *Level 1*, 29,% for *Level 2*, 42% for *Level 3*, 23% for *Level 4* and 5% for *Level 5*. It follows that *Level 1* and *Level 5* are missing enough information, and the distribution is skewed. Usually, imbalanced data is given, but having a multi-class classification with imbalanced data is even more daunting.

A popular way to treat this unequal distribution of classes is using SMOTE (*Synthetic Minority Over-sampling Technique*) [34]. Of course, there are other approaches, like under-sampling the majority class, but in this case, the model may lack useful examples. One can choose to use random over-sampling, but this strategy comes with its drawbacks. Random over-sampling simply recreates examples for the minority class without adding variety or useful information about those new objects, but it will increase the likelihood of the model overfitting. At the same time, SMOTE adds new samples in the minority class generating synthetic data.

The algorithm behind SMOTE implies searching for K-nearest neighbours of each minority item, then one of these neighbours will be randomly selected and, using linear interpolation, a new minority instance is produced.

SMOTE will target the minority class to generate synthetic points to almost match the number of data points in the majority class. Now as the synthetic data being generated is chosen from lines connecting the existing points (feature data) so we are basically incorporating less noise. Thus reducing the chances of overfitting. Generally, SMOTE is used to overcome the problem of overfitting, but it might also lead to overfitting as it is synthetic data and not real data.

The technique specified by SMOTE can be easily applied to a multi-class problem because the algorithm identifies the minority class against the remaining ones with the approach One-versus-All. Python has a support library for SMOTE algorithm, but the drawback is that it can be used only with continuous features, whereas the dataset used in this study is a mix of categorical and

Table 7
LR- SMOTE - measurements per classes.

Classes	Recall	Precision	Specificity	ROC-AUC
Level 1	0.42	0.08	0.95	0.69
Level 2	0.55	0.65	0.87	0.71
Level 3	0.59	0.67	0.79	0.69
Level 4	0.64	0.59	0.87	0.75
Level 5	0.42	0.24	0.97	0.67

Table 8
NN Sequential - SMOTE - measurements per classes.

Classes	Recall	Precision	Specificity	ROC-AUC
Level 1	0.35	0.14	0.98	0.66
Level 2	0.65	0.66	0.96	0.75
Level 3	0.63	0.68	0.78	0.70
Level 4	0.67	0.62	0.89	0.78
Level 5	0.39	0.33	0.95	0.67

Table 9
RFT - SMOTE - measurements per classes.

Classes	Recall	Precision	Specificity	ROC-AUC
Level 1	0.24	0.16	0.98	0.61
Level 2	0.60	0.58	0.74	0.71
Level 3	0.57	0.63	0.75	0.66
Level 4	0.61	0.54	0.85	0.73
Level 5	0.29	0.31	0.96	0.63

continuous attributes. Thus, we have used SMOTE-NC (SMOTE for *Nominal and Continuous features*) instead, and we have specified which are the *Categorical* features so that the SMOTE algorithm would resample these values instead of generating synthetic data. After synthetic samples were produced, the distribution was completely changed, and the amount of data was substantially increased to 188242 for each level.

All three models were re-evaluated using the same metrics to have an overview and to continue the study because we wanted to see how the models performed with the new samples. All the hyper-parameters were kept constant.

Again, the Neural Network model had a better performance in terms of ROC-AUC score than the previous Neural Network (Table 4), with an average of 0.71. The *recall* and *precision* of *Level 2, 3* and *4* were close like usual. The confusion matrix showed that most patients who were classified as non *Level 2* when they should have been assigned with that code were on *Level 3* and *4*. For *Level 3*, many patients were spread on *Level 2* and *Level 4*.

Out of all three models, *Levels 1,3,4* and *5* had the lowest *recall* score using Random Forest Tree model. Also, the ROC-AUC scores were lower than the previous ones depicted in Table 5. If we calculate the average on the *recall* metric, the result is the same as the one from the other RFT classifier.

In the end, according to Logistic Regression measurements, there was a higher *recall* for *Level 1* and *Level 5* comparing the results to the ones from Table 6. This time, a lot of patients were misclassified as high-risk (*Level 1*) patients. The ROC-AUC score was better than the previous Logistic Regression model using imbalanced data, but the *recall* and *precision* were lower.

In Machine Learning, more data usually means better predictions, but in the previous tests, it is clear that the differences are not that significant. Hence, we further check if we can get better predictions by augmenting the given dataset and we organize the results in Table 7, Table 8, and Table 9, respectively.

4.3. ADASYN for imbalanced data

ADASYN (Adaptive Synthetic) is an improved sampling technique than SMOTE, by reducing the bias of skewed data and sampling more data for the highly imbalanced classes that can be tough to learn by a model [35].

After we applied the ADASYN method to our imbalanced data, the number of samples was increased on each emergency level, but there were some minor differences between them as described in Fig. 9. In SMOTE case, the number of samples was the same for each class.

We evaluated all three models using that new dataset. The results were not far from the SMOTE experiment but slightly better.

Based on the results from Table 10, using the Logistic Regression model as in the case of SMOTE experiment, for *Level 1* and *Level 5* the emergency codes still had a high recall and a low precision, meaning the predictions were incorrect for this class.

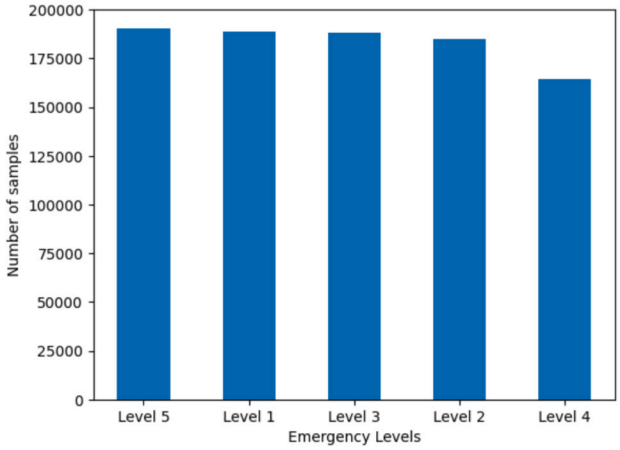


Fig. 9. ADASYN SAMPLING.

Table 10
LR- ADASYN - measurements per classes.

Classes	Recall	Precision	Specificity	ROC-AUC
Level 1	0.73	0.05	0.89	0.81
Level 2	0.49	0.60	0.86	0.68
Level 3	0.52	0.71	0.84	0.68
Level 4	0.53	0.62	0.90	0.72
Level 5	0.68	0.23	0.88	0.78

Table 11
NN Sequential - ADASYN - measurements per classes.

Classes	Recall	Precision	Specificity	ROC-AUC
Level 1	0.47	0.14	0.92	0.72
Level 2	0.66	0.64	0.84	0.75
Level 3	0.55	0.71	0.83	0.69
Level 4	0.58	0.62	0.89	0.74
Level 5	0.65	0.26	0.90	0.78

Table 12
RFT - ADASYN - measurements per classes.

Classes	Recall	Precision	Specificity	ROC-AUC
Level 1	0.25	0.27	0.99	0.62
Level 2	0.62	0.59	0.81	0.72
Level 3	0.61	0.62	0.73	0.67
Level 4	0.59	0.59	0.88	0.73
Level 5	0.31	0.41	0.97	0.64

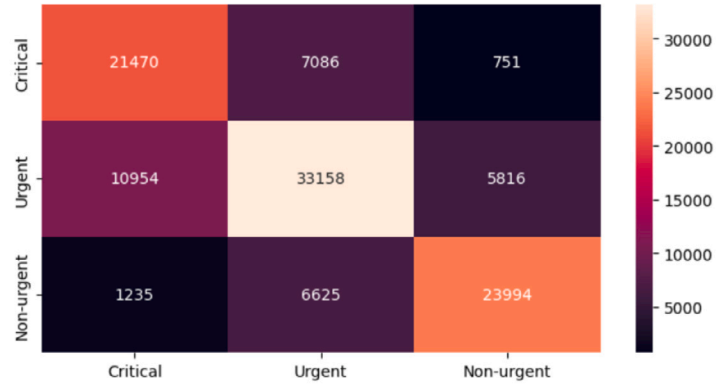
In the case of the NN Sequential Model (see Table 11), there was an improved score of ROC-AUC, precision and recall for levels 2, 3, and 4. However, the specificity was lower.

After evaluating the Random Forest Tree model, we observe in Table 12 that ROC-AUC for levels 2, 3, and 4 was even lower, meaning the model was not capable of distinguishing between classes correctly.

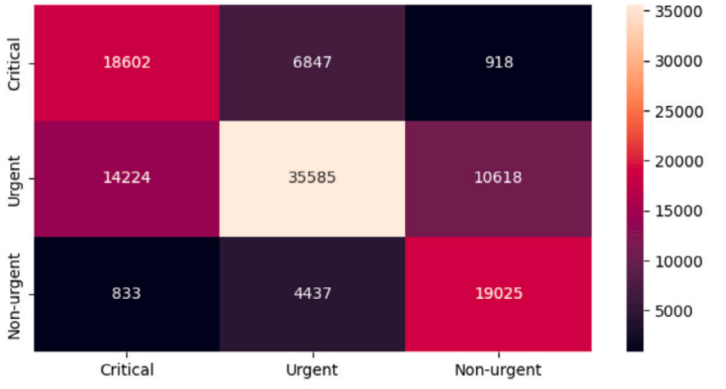
4.4. The simplified problem

In search of even better predictions, another approach is considered here. Instead of having a 5-level classification, we have assumed a 3-level classification problem. We simplified the problem to the point where the model would predict only three emergency situations by grouping *Level 1* and *Level 2* into one class called `Critical`, *Level 3* standing to represent the `Urgent` class and grouping again *Level 4* and *Level 5* as `Non-urgent` class. Basically, everything above *Level 3* (higher levels) represents a critical situation which,

(a) Simplified problem - NN Sequential model - confusion matrix



(b) Simplified problem - RFT model - confusion matrix



(c) Simplified problem - LR model - confusion matrix

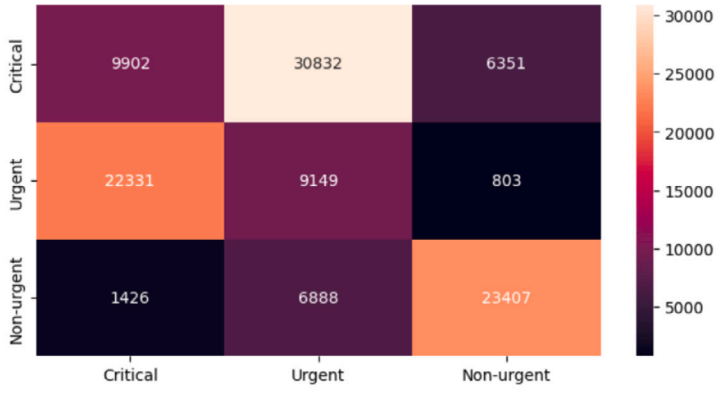


Fig. 10. Simplified problem - models - confusion matrix.

indeed, has to have maximum priority, and what is below this level (lower levels) represents a non-urgent case. In this manner, the classes were distributed naturally, following the organic meaning. The distribution of the new target features resulted as follows: `Critical` - 30%, `Urgent` - 43%, `Non-urgent` - 27%.

For the interest of this study, we kept the previous three models and the metrics as well for experiments in order to see how this perspective changes the algorithms' performance. The corresponding confusion matrix is resumed in Fig. 10 for all models, and Table 13 outlines the related measurements for the selected metrics.

Experiment 1 was an NN-Sequential approach providing that, for patients with an urgent emergency, the number of TP is 33158, whereas 13711 were wrongly assigned to another class. The amount of FN is pretty high. Looking at the measurements presented in Table 13, the *recall* of the `Urgent` class is slightly better than `Critical` one, but there is a lot of confusion between these two classes. The `Non-urgent` class has the best *recall*, followed by the `Urgent` one, which is the majority class. Although the `Critical`

Table 13
Simplified problem - measurements per classes.

Model	Class	Recall	Precision	Specificity	ROC-AUC
NN-Sequential	`Critical`	0.63	0.73	0.89	0.76
	`Urgent`	0.70	0.66	0.73	0.72
	`Non-urgent`	0.78	0.75	0.90	0.84
Random Forest Tree	`Critical`	0.55	0.70	0.89	0.72
	`Urgent`	0.75	0.58	0.61	0.67
	`Non-urgent`	0.62	0.78	0.93	0.77
Logistic Regression	`Critical`	0.29	0.21	0.51	0.40
	`Urgent`	0.19	0.28	0.63	0.41
	`Non-urgent`	0.76	0.73	0.69	0.83

class has the lowest *recall*, the *precision* is high. The ROC-AUC shows that the model is performing very well when predicting the `Non-urgent` class. The model predicted 78% of patients to be assigned on the `Non-urgent` class with a *precision* of 75% and predicted 90% as true negatives.

Experiment 2 applies the Random Forest Tree algorithm with the same number of estimators. In this case, the `Urgent` class has a better TP value than the one generated by the NN-Sequential, but the values for the other two classes decreased. The *recall* is higher for the `Urgent` class than the others, but the *precision* and *specificity* are low, even if the ROC-AUC shows the model is not distinguishing very well this class in this case. The others have low *recall* but better results for the other metrics. We may conclude that the Neural Network for the simplified problem has better scores overall.

The last one, *Experiment 3*, is again based on the Logistic Regression model. Its confusion matrix included in Fig. 10 indicates a lot of FNs and TPs for the first two classes. Looking also at Table 13, this approach has the lowest scores, and it is performing poorly compared to the other two algorithms. Even if the `Non-urgent` class has good scores here, the highest risk class, the `Critical` one, for example, is more important and has the lowest scores.

Considering all of these measurements made using the given dataset with different distributions of classes shows that sometimes a specific model is performing well and other times not. Analysing different measurements, the dataset should have more significant examples so the models can better distinguish the defined classes `Critical`, `Urgent` and `Non-urgent`, which are easily confused. Looking only at the *recall* score, the NN-Sequential algorithm performed best in all cases.

Emergency care is an important decision point in a hospital, and the solutions provided by Artificial Intelligence could solve many challenges [4]. This is the reason there are papers in the literature that use different strategies or other Machine Learning algorithms applied in emergency departments [36]. For example, the aim of paper [11] (KTAS) was to analyse different models on predicting "Korean Triage and Acuity Scale [11]" levels. In that paper, Logistic Regression, Random Forest Tree and XGBoost algorithms were evaluated both on clinical data and text data using NLP (Natural Language Processing). The authors also integrated NLP in [14] (KATE) for achieving better prediction values, and in this particular study, only XGBoost algorithm was evaluated. Another approach to the emergency room department is to predict the patient's admission like researchers of this paper [21] (ED-Admission) did.

For the purpose of creating a context, the models used in this study are compared with the ones used in the papers mentioned above. To have a diversity of techniques, the models trained on the simplified problems were used for this comparison, calculating an average on each metric, with the claim that this study is notated as eUPU. For KTAS we chose the measurements evaluated on all data as well for KATE ("pediatric and adult patients in the gold set [14]" were used in the latter case). The ED-Admission paper studied a binary problem, and we chose the evaluation of the models on all data. The diversity of algorithms and metrics, as well as scores, is represented in Table 14.

The classification performance and inferences of Machine Learning models could be improved by applying other methods and approaches, such as information on hyper-parameter calibration to make models more extensible and different methods to identify predictors that could affect the triage workflow. Modelling this problem is not the most complex and therefore, more powerful models can be proposed for this study (e.g. XGBoost [11], [14]). Also, uncertainty handling methods would be of interest in this particular case.

4.5. Memory and CPU usage

When dealing with Machine Learning algorithms, memory and CPU usage need to be evaluated. Although the accuracy of a model requires the main focus, execution time and memory increment should be also considered and evaluated.

For this topic, we utilized specific Python commands that are built into IPyhton that runs on Jupyter Notebook. To evaluate the CPU usage of *fit* and *predict* we used the command *%%time*, and to evaluate the memory usage, we used *%%memit*, which required a specific package, *memory_profiler*.

According to Fig. 11 and Fig. 12, on one hand, Logistic Regression used on samples made by SMOTE technique had a significant memory increment when fitting the training data. On the other hand, the NN-Sequential model required more CPU usage than the others. In that particular case, the dataset resulting from ADASYN technique had the highest execution time. When the *predict* was executed, models did not require as much memory and the execution time was fast.

Table 14
Models and metrics comparison.

Study	Model	Metric	Score
eUPU	NN-Sequential	ROC-AUC	0.76
		Precsion	0.70
		Recall	0.69
		Specificity	0.84
	Logistic Regression	ROC-AUC	0.54
		Precision	0.65
		Recall	0.41
		Specificity	0.61
	Random Forest Tree	ROC-AUC	0.72
		Precision	0.68
		Recall	0.64
		Specificity	0.81
KTAS	Logistic Regression	Precision	0.71
		Recall	0.70
		F1-score	0.70
		AUROC	0.90
	Random forest	Precision	0.73
		Recall	0.73
		F1-score	0.70
		AUROC	0.92
	XGBoost	Precision	0.75
		Recall	0.75
		F1-score	0.74
		AUROC	0.92
KATE	XGBoost	AUC	0.84
		F1-score	0.73
		Sensitivity	0.69
		Precision	0.80
ED-Admission	Logistic Regression	AUC	0.90
		Sensitivity	0.80
		Specificity	0.85
		PPV	0.69
		NPV	0.91
	XGBoost	AUC	0.92
		Sensitivity	0.83
		Specificity	0.85
		PPV	0.70
		NPV	0.92
	DNN	AUC	0.92
		Sensitivity	0.82
		Specificity	0.85
		PPV	0.70
		NPV	0.92

5. Interfacing by the web-based application

The challenging research for finding a Machine Learning model well performing in emergency department triage flow led us to develop an interface between the model and the clinician as the end-user. A series of software resources have been selected for this approach, successfully guided by the Python [17] ability to export the Machine Learning model as an API (*Application Programming*

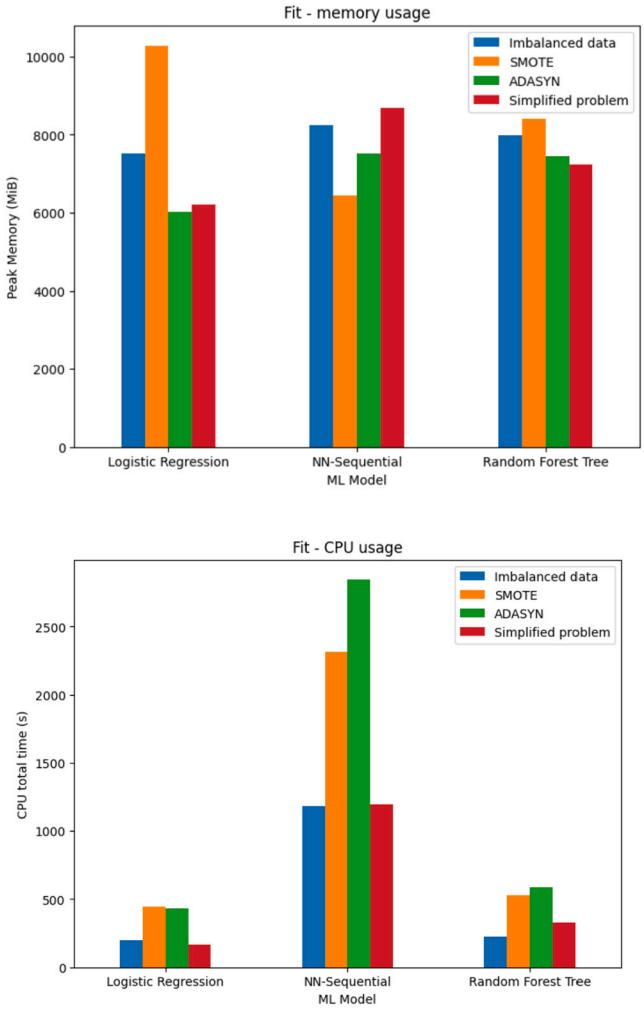


Fig. 11. Fitting data - Memory and CPU usage.

Interface) using Flask [37]. The end-user interface was built with React technology [38] for getting the predictions of the trained model using the API provided by Flask, following the application architecture represented in Fig. 13.

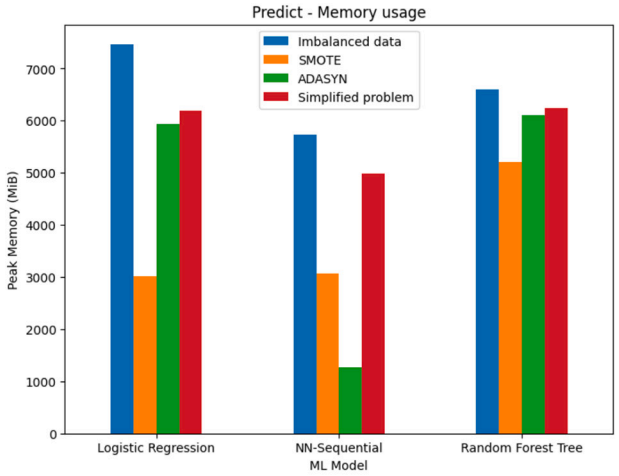
The Machine Learning model was saved using the Python *joblib* package and imported as an instance in Flask app. We chose to integrate the NN-Sequantial model trained on the simplified problem due to average scores described in Table 14. The same preprocessing transformers described in Section 3 and Section 4 have been applied and after the appropriate dataset was preprocessed, the imported classifier would make predictions. Both the simple data flow and the user-friendly interface facilitate the interaction with clinicians, keeping the main focus on the performance of the used Machine Learning algorithm.

Making the ML-based application available and applicable to the various domains allows people with high expertise in those domains, here medical care, to accept the IT tools for supporting a more reliable clinical decision.

6. Discussion and conclusions

This research study focuses on a Machine Learning approach to assign emergency codes to each patient after triage. We demonstrated the potential of an automatic classification system using supervised models to enhance early medical diagnosis and facilitate medical record processing. AI applications could significantly enhance the efficiency of the emergency triage process. Emergencies need to be prioritized, so studying these models in real-time and evaluating the dynamic power of machine learning for classifying which emergency code should a patient receive is entirely justified.

Both the Machine Learning algorithms and the medical domain turned out to be more challenging than expected. Medical data is not easy to find, and it has to be customized by country, e.g. Romanian patient file differs from the one used in the USA. Therefore, we also targeted to adjust the dataset to the Romanian medical care system. Also, we had to take care of missing data, and we were surprised to see how imbalanced the classes were, but the aim was to study the dataset and see how it could be applied in this



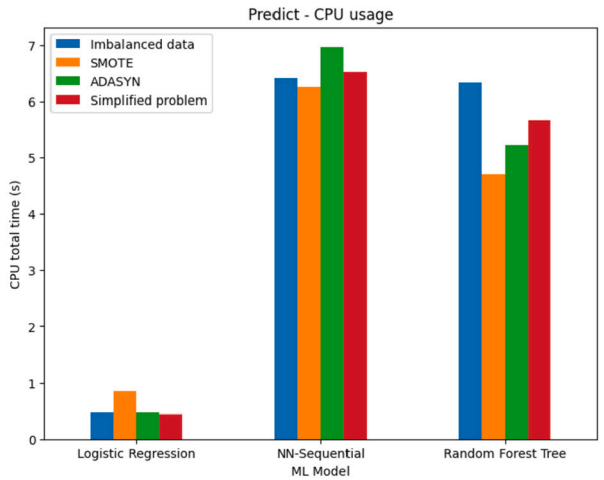


Fig. 12. Predicting data - Memory and CPU usage.

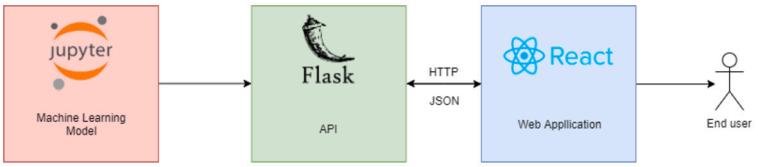


Fig. 13. Application Architecture.

environment. Eventually, we simplified the problem due to imbalanced data and kept the algorithm with the best results. After our research, the experimental results demonstrated that the NN-Sequential algorithm performed better on the simplified problem.

Without any doubt, there are opportunities to improve this study, and there is further work to do, but for this, more data needs to be collected. Besides this, good quality feedback from experts in that particular domain is needed, and this can be achieved by our web application, where we use the best classifier as an API. The tool itself proves easy to use as an interface for patient overview and early monitoring, and our findings indicate that clinicians were able to reliably use the tool. In the end, having impressive accuracy in predicting emergency codes, a digital smart patient file could be created.

Ethics statement

We hereby confirm that our study complies with all regulations and we correspondingly cited as [21] the source of our dataset with respect to the original source [20] that mentions: *We provide the de-identified dataset and R scripts for the paper "Predicting hospital admission at emergency department triage using machine learning". All processing scripts in the Scripts/R/ subdirectory take as input .csv files extracted from the enterprise data warehouse using SQL queries. The analysis scripts in the main directory take as input the de-identified*

dataset provided in this repository. The working directory should be set to the main directory with the analysis scripts. All research using this dataset should cite the original paper. "Hong WS, Haimovich AD, Taylor RA (2018) Predicting hospital admission at emergency department triage using machine learning. PLoS ONE 13(7): e0201016.".

CRediT authorship contribution statement

Andreea Vantu: Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.

Anca Vasilescu: Conceived and designed the experiments; Performed the experiments; Contributed reagents, materials, analysis tools or data; Wrote the paper.

Alexandra Baicoianu: Conceived and designed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data included in article/supp.material/referenced in article.

References

[1] E. Alpaydin, Introduction to Machine Learning, fourth edition, Adaptive Computation and Machine Learning Series, MIT Press, 2020.
[2] J. Alzubi, A. Nayyar, A. Kumar, Machine learning from theory to algorithms: an overview, in: Second National Conference on Computational Intelligence, NCCI 2018, 5 December 2018, Bangalore, India, J. Phys. Conf. Ser. 1142 (012012) (2018), https://doi.org/10.1088/1742-6596/1142/1/012012.
[3] P. Sharma, Z. Said, A. Kumar, S. Nizetic, A. Pandey, A. Hoang, Z. Huang, A. Afzal, C. Li, T. Le Anh, X.P. Nguyen, V. Tran, Recent advances in machine learning research for nanofluid-based heat transfer in renewable energy system, Energy Fuels 36 (13) (2022) 6626–6658, https://doi.org/10.1021/acs.energyfuels.2c01006.
[4] Kenneth Jian Wei Tang, Candice Ke En Ang, T. Constantinides, V. Rajinikanth, U. Rajendra Acharya, K.H. Cheong, Artificial intelligence and machine learning in emergency medicine, Biocybern. Biomed. Eng. 41 (1) (2021) 156–172, https://doi.org/10.1016/j.bbe.2020.12.002.
[5] R. Sánchez-Salmerón, J.L. Gómez-Urquiza, L. Albendín-García, M. Correa-Rodríguez, M.B. Martos-Cabrera, A. Velando-Soriano, N. Suleiman-Martos, Machine learning methods applied to triage in emergency services: a systematic review, Int. Emerg. Nurs. 60 (2022) 101109, https://doi.org/10.1016/j.ienj.2021.101109.
[6] C.K. Wee, X. Zhou, R. Sun, R. Gururajan, X. Tao, Y. Li, N. Wee, Triaging medical referrals based on clinical prioritisation criteria using machine learning techniques, Int. J. Environ. Res. Public Health 19 (12) (2022) 7384, https://doi.org/10.3390/ijerph19127384.
[7] G.D. Magoulas, A. Prentza, Machine learning in medical applications, in: G. Paliouras, V. Karkaletsis, C.D. Spyropoulos (Eds.), Machine Learning and Its Applications, ACAI 1999, in: Lecture Notes in Computer Science, vol. 2049, Springer, Berlin, Heidelberg, 2001.
[8] M. Shehab, L. Abualigah, et al., Machine learning in medical applications: a review of state-of-the-art methods, Comput. Biol. Med. 145 (June 2022), https://doi.org/10.1016/j.compbiomed.2022.105458, 2022.
[9] A. Mujumdar, V. Vaidehi, Diabetes prediction using machine learning algorithms, Proc. Comput. Sci. 165 (1) (2019) 292–299, https://doi.org/10.1016/j.procs.2020.01.047.
[10] S. Levin, M. Toerper, et al., Machine-learning-based electronic triage more accurately differentiates patients with respect to clinical outcomes compared with the emergency severity index, Ann. Emerg. Med. 71 (5) (September 2017), https://doi.org/10.1016/j.annemergmed.2017.08.005, 2017.
[11] S.W. Choi, T. Ko, K.J. Hong, K.H. Kim, Machine learning-based prediction of Korean triage and acuity scale level in emergency department patients, J. Phys. Conf. Ser. 25 (4) (2019) 305–312, https://doi.org/10.4258/hir.2019.25.4.305.
[12] A. Vântu, Using technology to improve the emergency room triage, Bachelor Thesis supported by Siemens Romania SRL under the contract BCT 25101/10.04.2017, Transilvania University of Braşov, Romania, June 2018.
[13] A. Vântu, A. Vasilescu, e-UPU: using technology to improve the emergency room triage, in: Poster Session, 7th ACM Celebration of Women in Computing: womENcourage 2020, ADA University, Baku, Azerbaijan, 24-27 September 2020, also Available Online, 2020.
[14] O. Ivanov, L. Wolf, D. Brecher, E. Lewis, K. Masek, K. Montgomery, Y. Andrieiev, M. McLaughlin, S. Liu, R. Dunne, K. Klauer, C. Reilly, Improving ED emergency severity index acuity assignment using machine learning and clinical natural language processing, J. Emerg. Nurs. 47 (2) (2021) 265–278, https://doi.org/10.1016/j.jen.2020.11.001.
[15] N. Gilboy, et al., Emergency Severity Index (ESI) - a Triage Tool for Emergency Department Care, Implementation Handbook, Version 4, ENA Emergency Nurses Association, available online (2020).
[16] H. Harvey, How data scientists can convince doctors that AI works, available online.
[17] Python 3.9.1 documentation, https://docs.python.org/release/3.9.1/. (Accessed 26 August 2022).
[18] D.N. Lipe, S.S. Bourenane, M.K. Wattana, S. Gaeta, P. Chaftari, M.T. Cruz Carreras, J.-G. Manzano, C. Reyes-Gibby, A modified emergency severity index level is associated with outcomes in cancer patients with COVID-19, Am. J. Emerg. Med. 54 (2022) 111–116, https://doi.org/10.1016/j.ajem.2022.02.002.
[19] M.H. Yarmohammadian, F. Rezaei, A. Haghshenas, N. Tavakoli, Overcrowding in emergency departments: a review of strategies to decrease future challenges, J. Res. Med. Sci. 22 (2017) 23, https://doi.org/10.4103/1735-1995.200277.
[20] W.S. Hong, A.D. Haimovich, A.R. Taylor, Predicting hospital admission at emergency department triage using machine learning, https://github.com/yaleemmlc/admissionprediction, dataset available from 6 Sep 2018.
[21] W.S. Hong, A.D. Haimovich, R.A. Taylor, Predicting hospital admission at emergency department triage using machine learning, PLoS ONE 07 (13) (2018) 1–13, https://doi.org/10.1371/journal.pone.0201016.
[22] A. Géron, Hands on Machine Learning with Scikit-Learn, Keras, and TensorFlow, second edition, O'Reilly Media Inc., USA, 2019.
[23] scikit-learn Homepage, available online (Accessed 26 August 2022).
[24] E.C. Zabor, C.A. Reddy, R.D. Tendulkar, S. Patil, Logistic regression in clinical studies, Int. J. Radiat. Oncol. Biol. Phys. 112 (2) (2022) 271–277, https://doi.org/10.1016/j.ijrobp.2021.08.007.

[25] E. Boateng, D. Abaye, A review of the logistic regression model with emphasis on medical research, J. Data Anal. Inf. Process. 7 (2019) 190–207, https://doi.org/10.4236/jdaip.2019.74012.
[26] S. Dreiseitl, L. Ohno-Machado, Logistic regression and artificial neural network classification models: a methodology review, J. Biomed. Inform. 35 (5–6) (2002) 352–359, https://doi.org/10.1016/S1532-0464(03)00034-0.
[27] M. Fernandez-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems?, J. Mach. Learn. Res. 15 (2014) 3133–3181, https://dl.acm.org/doi/10.5555/2627435.2697065.
[28] A. Lekhtman, Data Science in Medicine — Precision & Recall or Specificity & Sensitivity? Available online (Accessed 26 August 2022).
[29] D.G. Levy, In Machine Learning Predictions for Health Care the Confusion Matrix is a Matrix of Confusion, available online (Accessed 26 August 2022).
[30] B. Bowers, Triage to AI: a Machine Learning Approach to Hospital Admissions Classification, available online (Accessed 26 August 2022).
[31] S.E. Awan, M. Bennamoun, F. Sohel, F.M. Sanfilippo, G. Dwivedi, Machine learning-based prediction of heart failure readmission or death: implications of choosing the right model and the right metrics, ESC Heart Fail. 6 (2) (2019) 428–435, https://doi.org/10.1002/ehf2.12419.
[32] Understanding medical tests: sensitivity, specificity, and positive predictive value, available online (Accessed 16 January 2021).
[33] B. Krawczyk, Learning from imbalanced data: open challenges and future directions, Prog. Artif. Intell. 5 (4) (2016) 221–232, https://doi.org/10.1007/s13748-016-0094-0.
[34] A. Fernández, S. Garcia, F. Herrera, N.V. Chawla, SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary, J. Artif. Intell. Res. 61 (1) (January 2018) 863–905, https://doi.org/10.1613/jair.1.11192, 2018.
[35] H. He, Y. Bai, E.A. Garcia, S. Li, ADASYN: adaptive synthetic sampling approach for imbalanced learning, in: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, 2008, pp. 1322–1328, https://doi.org/10.1109/IJCNN.2008.4633969, 2008.
[36] B. Wang, W. Li, et al., Improving triaging from primary care into secondary care using heterogeneous data-driven hybrid machine learning, Decision Support Systems 166, https://doi.org/10.1016/j.dss.2022.113899, 2023.
[37] Flask's documentation, https://flask.palletsprojects.com/en/2.2.x/. (Accessed 26 August 2022).
[38] React Getting Started, https://reactjs.org/docs/getting-started.html. (Accessed 26 August 2022).