2022 年臺灣國際科學展覽會
優勝作品專輯

作品編號	**190042**
參展科別	電腦科學與資訊工程
作品名稱	**Development of an Android Application for Triage Prediction in Hospital Emergency Departments**
得獎獎項	

國　　家	**Philippines**
就讀學校	**Philippine Science High School–Main Campus**
指導教師	**Donna Salve C. Hipolito**
作者姓名	**Clyde Ambroz S. Acyatan** **Lucas Sebastian F. Khan** **Uriel Nathan D. Orpilla**

關鍵詞　**triage、machine learning、Android**

作者照片

ABSTRACT

Triage is the process by which nurses manage hospital emergency departments by assigning patients varying degrees of urgency. While triage algorithms such as the Emergency Severity Index (ESI) have been standardized worldwide, many of them are highly inconsistent, which could endanger the lives of thousands of patients. One way to improve on nurses' accuracy is to use machine learning models (ML), which can learn from past data to make predictions. We tested six ML models: random forest, XGBoost, logistic regression, support vector machines, k-nearest neighbors, and multilayer perceptron. These models were tasked with predicting whether a patient would be admitted to the intensive care unit (ICU), another unit in the hospital, or be discharged. After training on data from more than 30,000 patients and testing using 10-fold cross-validation, we found that all six models outperformed ESI. Of the six, the random forest model achieved the highest average accuracy in predicting both ICU admission (81% vs. 69% using ESI; $p < 0.001$) and hospitalization (75% vs. 57%; $p < 0.001$). These models were then added to an Android application, which would accept patient data, predict their triage, and then add them to a priority-ordered waiting list. This approach may offer significant advantages over conventional triage: mainly, it has a higher accuracy than nurses and returns predictions instantaneously. It could also stand-in for triage nurses entirely in disasters, where medical personnel must deal with a large influx of patients in a short amount of time.

INTRODUCTION

Background

Hospital overcrowding has become a healthcare crisis in many nations (Pines et al., 2011), especially during the ongoing COVID-19 pandemic. To manage their emergency departments (EDs), hospitals use triage, a method of assigning varying degrees of urgency to different patients based on the judgement of nurses. Every day, nurses must make life-or-death decisions when triaging different patients. The incorrect triage of a patient may result in either giving a higher priority to a less serious problem or giving a lower priority to a more serious problem. In some cases, an incorrect triage may cost the life of a patient. When using the Emergency Severity Index (ESI), which is the most used triage system in the U.S., nurse triage accuracy is approximately 60% (Emergency Nurses' Association, n.d.) which means that there may be millions of patients being incorrectly triaged.

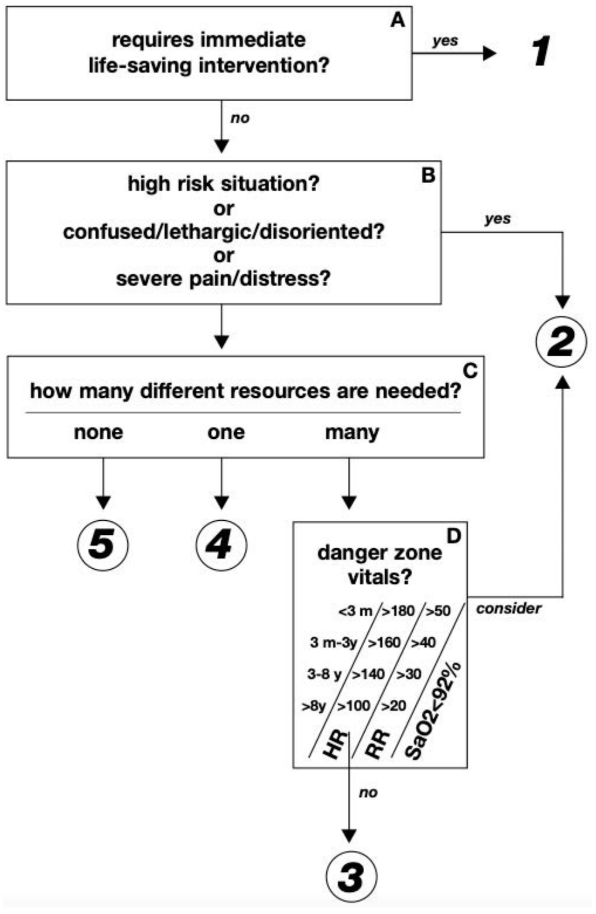

A
requires immediate life-saving intervention?
yes
1
no
B
high risk situation? or confused/lethargic/disoriented? or severe pain/distress?
yes
2
C
how many different resources are needed?
none
one
many
5
4
D
danger zone vitals?
<3 m
>180
>50
3 m-3y
>160
>40
3-8 y
>140
>30
>8y
>100
>20
HR
RR
SaO2<92%
consider
no
3

Figure 1. The procedure for the Emergency Severity Index, which gives patients a triage rating from 1 (most severe) to 5 (least severe). Taken from Gilboy et al. (2011).

This inaccuracy is often because triage is often reliant on nurses' subjective judgement of pain and critical care needs, as seen in Step B of Figure 1. Additionally, the highly procedural nature of triage, which involves distilling a multitude of factors into a single numerical rating, may paint an incomplete picture of a patient's condition. This could mean that even if a triage score is procedurally correct, it may overcompensate for or not meet a patient's needs. For these reasons, triage may have high variation and low reliability (Kwon et al., 2018).

One commonly proposed tool to aid nurses in triage is machine learning. Machine learning is commonly proposed for use as a tool in triage as it fixes many of its flaws concerning subjectivity or complexity. Unlike triage nurses, it has the benefit of hindsight: the final status of each patient (such as deceased or hospitalized), instead of triage score, is used to train the model. From here, it can detect trends in the data that led to various diagnoses.

Objectives

The specific objective of this study was to create machine learning models that achieve a significantly higher accuracy than nurses using the Emergency Severity Index in predicting both intensive care unit (ICU) admission and hospitalization. Our main objective, then, would be to implement the most accurate model in an Android application which would accept patient data, use it to predict their triage classification, and then add patients to a waiting list ordered according to priority. This allows the efficient management of patients in the emergency department while also allowing patients to see how long they must wait before being treated.

Significance of the Study

This project could help to reduce overcrowding in hospital emergency departments. Our application could classify patients with a level of accuracy comparable to or greater than that of nurses, and its ability to generate a prediction in a matter of seconds gives it a significant speed advantage. It could also be especially useful in emergency dispatch centers or at the scene of a large-scale emergency or disaster, where medical personnel must deal with a large influx of patients in a very short amount of time.

REVIEW OF RELATED LITERATURE

Machine learning

Artificial intelligence (AI) has become increasingly ubiquitous in the modern world, powering everything from web searches on the internet to user-tailored recommendations on e-commerce sites (LeCun et al., 2015). Unlike a normal computer program, which after a thousand runs that produce a wrong answer does not re-educate its handiwork (Michie, 1968) software that implements AI technology is able to learn on its own without being given explicit instructions on how to do so (Silver et al., 2017).

Machine learning is a subset of artificial intelligence concerned with the study of algorithms that “improve automatically through experience” (Mitchell, 1997). These algorithms can learn from large amounts of sample data, from which it creates a mathematical model capable of making predictions or decisions.

Machine learning in triage classification

Machine learning has a variety of applications in various fields where solving some problems using conventional algorithms can be inaccurate, impractical, or even impossible. One such algorithm is triage. Researchers have created models with accuracy rates significantly greater than that of nurses, especially in predicting specific *outcomes*, such as intensive care unit (ICU) admission. Algorithms such as random forests and logistic regression have achieved higher true positive rates, and lower false negative rates, than triage systems such as the Modified Early Warning System (MEWS) and the Korean Triage and Acuity Scale (KTAS). However, deep learning algorithms have, in many studies, outperformed all of these (Kwon et al., 2018; Raita et al., 2019). Possible reasons behind the success of these algorithms will be discussed in the next sections.

Data preprocessing and data mining

The success of the aforementioned triage prediction algorithms was highly dependent on the quality and quantity of their training data. Triage data on hundreds of thousands of patients is currently available for use from sources such as the U.S. Center for Disease Control and Prevention (Raita et al., 2019). However, to ensure the quality and consistency of training data, it must be preprocessed before being fed to a model. If data is not preprocessed, the predictions made by the machine learning model could be incorrect or unreliable.

Data cleaning

Data cleaning is a method of data preprocessing wherein data points that have null or incomplete values are removed (Chu et al., 2016). However, if a dataset is too small, null values may be replaced by the mean (for more homogenous datasets) or median (for skewed datasets) value of its corresponding variable (Elragby, 2019). This increases the accuracy of the machine learning model's prediction as it ensures the quality of the data being fed to the model.

Feature engineering

Data mining is a method of finding patterns in datasets. Feature engineering is a method of data mining that uses general information about the target population to modify or create new dependent variables. This process is best understood in the context of a problem: for example, in making their triage classifier, Hong et al. (2018) edited their dataset's recording of patients' chief complaints, which initially had more than 1000 unique conditions. The top 200 most frequent complaints, which comprised more than 90% of the dataset, were kept, while the rest were marked as "Other". This made it easier for the machine learning model to mechanically determine patterns found inside the dataset – after all, it is easier to find a pattern when its repetitiveness, as well as its manner of repetitiveness, is distinguishable (Ng, 2013).

Classification algorithms

In machine learning and statistics, *classification* is the process of sorting datapoints into two or more specific categories. This is done by first splitting a dataset into a *training dataset* and a *testing dataset*. The training dataset contains sample datapoints whose categories are known; from this, a machine learning model would attempt to derive why each datapoint was in its corresponding category. After training, the model is tested by having it predict the category of each datapoint on the testing dataset.

Decision tree classifiers

Decision tree classifiers are a commonly used classification algorithm. They are predictive models that function by automatically generating decision trees. These are essentially flowcharts that gradually narrow down an object's possible category.

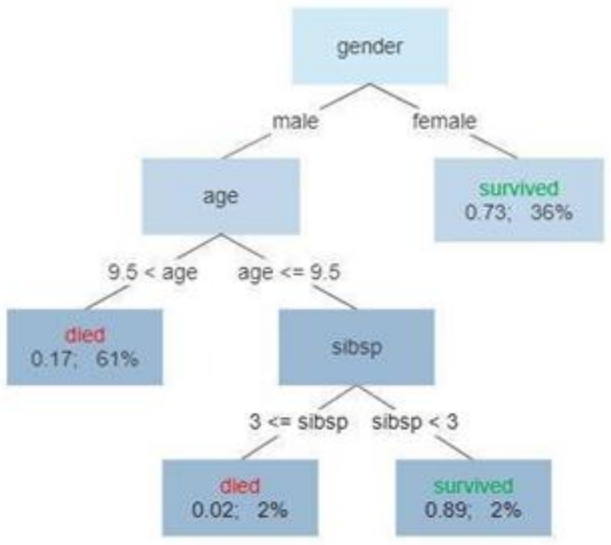


Figure 2. A basic decision tree for predicting the survival of passengers aboard the Titanic. ("sibsp": number of siblings) (Wikimedia Commons, 2020)

Decision tree learning uses the divide-and-conquer algorithm (Wu et al., 2008), wherein the "flowchart" of sorts recursively branches out until doing so no longer enhances the decision tree's predictive accuracy. This mechanism provides an insight into exactly how the model classifies patients, which could be very useful in the automation of procedural tasks such as triage.

A more complex and accurate form of decision trees are *ensemble* classifiers, which generate more than one decision tree. This literature review will discuss two ensemble methods: gradient boosted trees (XGBoost) and random forests. Gradient boosted decision trees create many weak decision trees and sequentially combine them, progressively making more accurate trees (Wu et al., 2008). Meanwhile, random forests operate by creating a multitude of decision trees and returning the most frequent prediction (Ho, 1995).

Logistic regression

Univariate logistic regression

The logistic regression algorithm uses the **logistic function** to perform binary classification. It is defined by the equation

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

In its simplest form, the algorithm is tasked with predicting the **category** of a single numeric variable t. Since the logistic function produces a value between 0 and 1 for all values of t, its predicted category is equal to $\sigma(t)$, rounded up or down (Sperandei, 2014). If $\sigma(t) < 0.5$, the variable t belongs to category 0, and if $\sigma(t) \geq 0.5$, it belongs to category 1.

Multivariate logistic regression

To make predictions from large datasets, the algorithm represents each variable in the dataset as a feature vector x_i. Each feature vector is multiplied to a weight vector w_i to produce a score t:

$$t = \sum_{n=1}^{i} w_1 x_1 + w_2 x_2 + \cdots w_i x_i$$

After hundreds, or sometimes thousands, of iterations, the weights, which start as random values, are optimized to produce the most accurate result. The score t is then passed through the logistic function, which returns a datapoint's category (0 or 1).

Neural networks

Neural networks are essentially layers of logistic regression classifiers stacked on top of each other. In a neural network, each neuron takes in a feature vector x_i from the previous layer and sends the dot product $w_i x_i$ to the next layer, which repeats the process (Castelvecchi, 2016). After a multitude of iterations, the weights are slowly optimized. And, as data progresses through the layers, the output of each neuron is calculated by taking the weighted sum of all its inputs and passing it through an activation function, such as $f(x) = \tanh x$. This produces an output between 0 (false) and 1 (true), equal to the probability that a datapoint will fall into one of the classes.

Support vector machines

Support-vector machines (SVMs) use a geometric approach to classification or regression. In it, each feature is represented as an axis on a graph: as such, a dataset with three features would correspond to a three-dimensional graph. Each datapoint is then plotted on the graph:

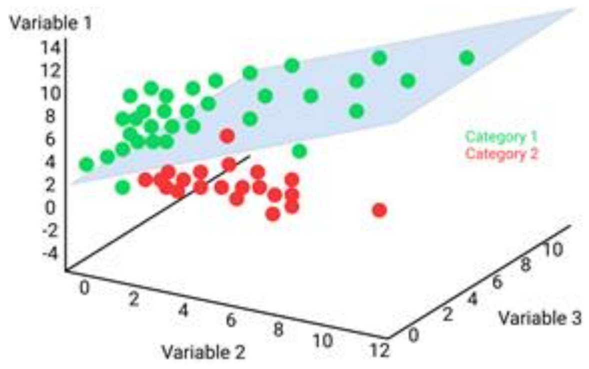


Figure 3. Support vector machine-based classification. Adapted from Noble (2006).

The datapoints are separated by a hyperplane, which is positioned to maximize the space between itself and any datapoints near it. The datapoints on each side of the hyperplane generally correspond to their own category; however, outliers may be dealt with using a “soft margin”. This is a user-specified parameter that controls, roughly, “how many examples are allowed to violate the hyperplane and how far across they are allowed to go” (Noble, 2006, p. 1566).

k-nearest neighbors

These models predict the category of a datapoint by calculating the proximity – usually Euclidean distance – between itself and every other datapoint. From here, the model classifies a datapoint based on the categories of its *k* nearest neighbors, with *k* being a parameter (typically a small, odd, positive integer) set by the user. The predicted class is the most frequently occurring class among these neighbors (Altman, 1992); however, in some variations, the "vote" of each neighbor is weighted according to its distance from the datapoint.

METHODOLOGY

Data preprocessing

This study made use of U.S. National Center for Health Statistics (2018) datasets, which contain data of ED patients collected during and after triage (see Appendix D). We combined 8 different datasets – one from each year from 2011 to 2018, with each having the same variables – and excluded patients who had missing data, were dead upon arrival at the ED, left before being seen, or left against medical advice, leaving a total of 151,383 patients. Of this number, 2284 (1.50%) of patients were admitted to the ICU, 14160 (9.35%) were admitted to another hospital ward, and 134,939 (89.1%) were discharged after receiving treatment. Using the Python library pandas, we then cleaned the data, a process described in the diagram below:

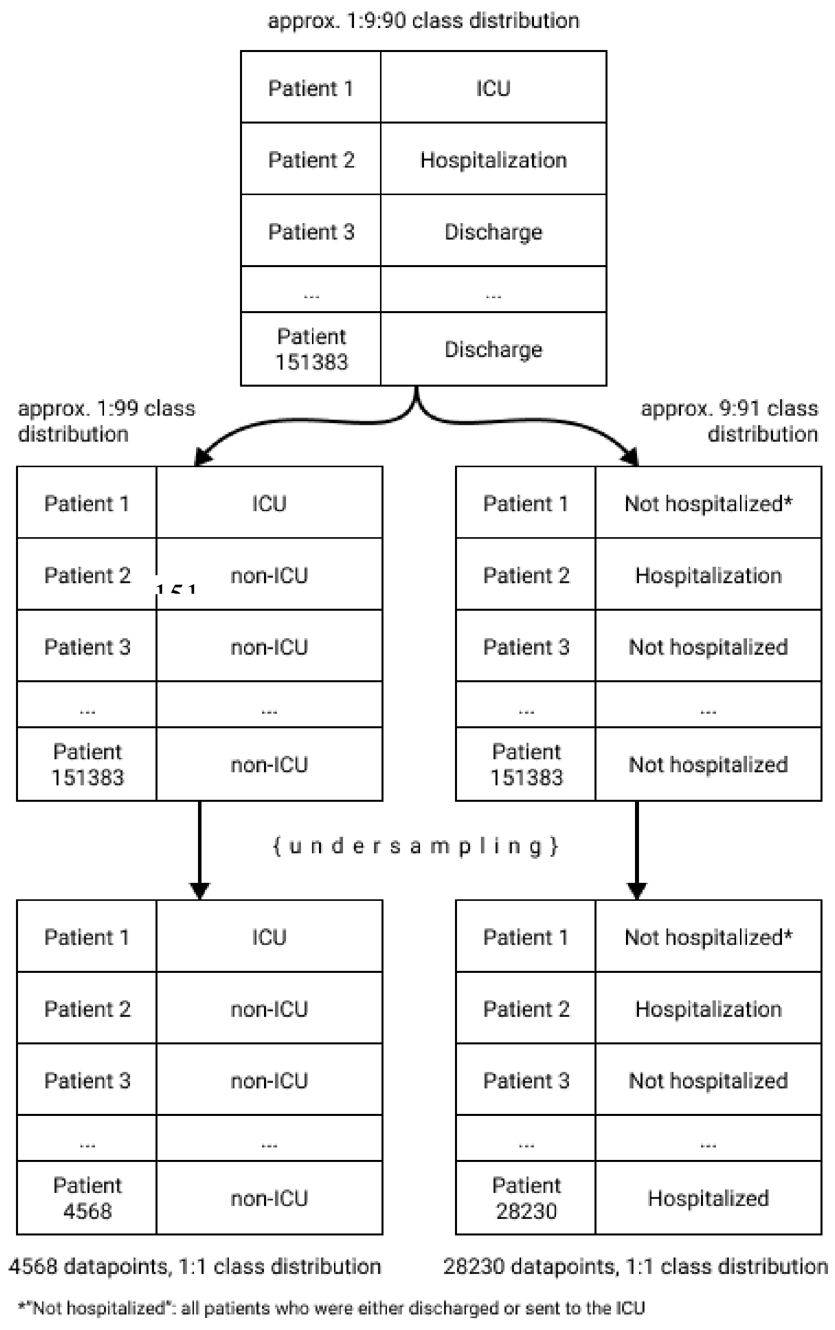
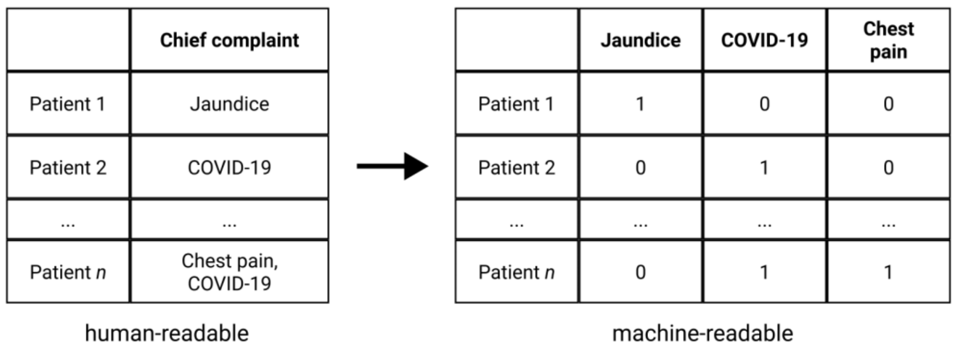


Figure 4. The dataset was modified to reduce classification complexity.

Feature Engineering

Each patient's age, sex, mode of arrival, initial vital signs, chief complaints, and prior chronic illnesses were used as predictor variables. Categorical variables, such as chief complaints, were one-hot encoded using the pandas get_dummies function:



	Chief complaint
Patient 1	Jaundice
Patient 2	COVID-19
...	...
Patient *n*	Chest pain, COVID-19

human-readable

	Jaundice	COVID-19	Chest pain
Patient 1	1	0	0
Patient 2	0	1	0
...	...	...	...
Patient *n*	0	1	1

machine-readable

Figure 5. One-hot encoding of categorical variables.

Training and evaluation of machine learning models

Six machine learning models were tested: random forest, XGBoost, logistic regression with Lasso regularization, support vector machines, *k*-nearest neighbors, and multilayer perceptron (neural network). The Python library scikit-learn was used to train and test all models. To maximize the models' accuracy, each one underwent grid-search hyperparameter tuning, which loops through all possible combinations of a model's settings to find the combination that yields the highest accuracy. Then, each model was trained and tested using 10-fold cross validation, from which their mean accuracy across all ten folds was calculated. The models were trained and tested on both datasets (see Figure 3). After testing, the most accurate model was ported (using the Python library sklearn-porter) to machine-readable Java code so that it could run on an Android application.

Coding of Android Application

The Android application was coded on Android Studio. It contains two Java classes, with one controlling the form input and the other controlling the list of patients. The first class receives user input when the user clicks a "submit" button and sends it to the next class. Upon receiving this information, the second Java class stores it in a data structure known as a priority queue, which sorts patients according to their triage rating.

RESULTS AND DISCUSSION

Prediction of ICU admission

As shown in Figure 4, each model that was tested substantially outperformed triage nurses in predicting intensive care unit (ICU) admission, with the random forest model performing best.

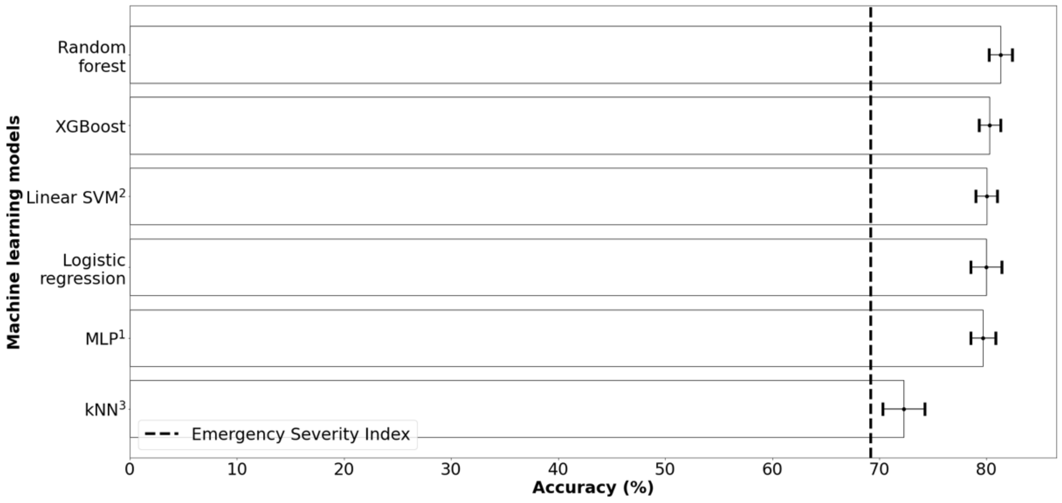


Figure 6. Mean accuracy (%) (95% CI) for ICU admission prediction by all six models, compared to the Emergency Severity Index (ESI). Accuracy was averaged across N=10 folds, with 456 or 457 test samples in each fold, totaling 4568 samples. All models were significantly more accurate than ESI ($p = 0.0024$ for *k*NN; $p < 0.001$ for all the others) (95% CL), although between models, none was significantly more accurate except between any of the first five and *k*-nearest neighbors.

[1] = Multilayer perceptron, [2] = Support vector machine, [3] = *k*-nearest neighbors

This significant discrepancy in accuracy between our models and nurses may be explained by data in Table 1:

Table 1. Mean accuracies, true negative rates, and true positive rates of all models, in addition to nurses using the Emergency Severity Index, in predicting ICU admission.

Models	Mean accuracy (%) (95% CI)	Mean true negative rate (%) (95% CI)	Mean true positive rate (%) (95% CI)
Random forest	81.44 (80.29-82.58)	78.91 (77.31-80.52)	83.90 (82.19-85.61)
XGBoost	80.19 (79.09-81.29)	79.76 (77.34-82.18)	80.72 (78.86-82.58)
MLP	80.36 (79.41-81.32)	78.87 (76.77-80.95)	81.85 (80.12-83.57)
Logistic regression	79.99 (78.29-81.69)	81.10 (78.72-83.48)	78.87 (77.21-80.53)
Linear SVM	79.82 (78.53-81.10)	81.32 (79.46-83.19)	78.32 (75.91-80.73)
*k*NN	72.50 (71.35, 73.66)	77.20 (74.83-79.58)	67.92 (65.61-70.22)
ESI	68.97	89.09	49.25

Table 1 shows that triage nurses (ESI) have a superior false negative rate: for every 100 patients who were *not* sent to the ICU, only 11 *should have* been sent to the ICU. However, nurses have a significantly worse false positive rate: for every 100 people they suggested to *send to the ICU*, approximately 49 – less than half – actually needed ICU treatment. This low true positive rate is alarming because it means that many patients who receive treatment at ICUs in the United States do not necessarily need it. This could needlessly lengthen the waiting time at ICUs, which could, by delaying treatment, cause some patients their lives.

Our most accurate model, meanwhile, triaged 83.90% (95% CI 82.19-85.61) of these patients correctly – significantly ($p < 0.001$; 95% CL) higher than nurses' true positive rate. The two decision tree models – gradient boosted trees (XGBoost) and random forest – were the most accurate models, with the random forest classifier being the more accurate of the two. This may be because triage itself can often come in the form of a decision tree – something that a random forest model, which builds multiple decision trees and chooses the most appropriate one, may be able to approximate.

Prediction of hospitalization

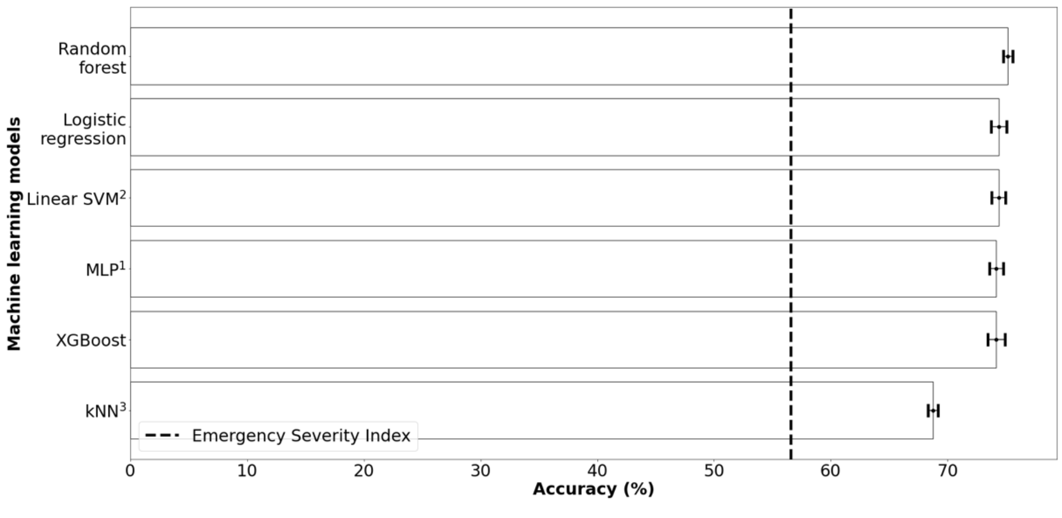


Figure 7. Mean accuracy (%) (95% CI) across 10 folds for prediction of hospitalization by all six models, compared to ESI. Accuracy was averaged across N=10 folds, with 2823 test samples in each fold, totaling 28230 samples. All models were significantly more accurate ($p < 0.001$; 95% CL) than nurses,

although between models, none was significantly more accurate except between any of the first five and *k*-nearest neighbors.

[1] = Multilayer perceptron, [2] = Support vector machine, [3] = *k*-nearest neighbors

Table 2. Mean accuracies, true negative rates, and true positive rates of all models, in addition to nurses using the Emergency Severity Index, in predicting hospitalization.

Models	Mean accuracy (%) (95% CI)	Mean true negative rate (%) (95% CI)	Mean true positive rate (%) (95% CI)
Random forest	75.18 (74.76-75.60)	73.59 (73.04-74.13)	76.78 (76.11-77.44)
XGBoost	74.17 (73.43-75.05)	75.96 (75.33-76.59)	74.33 (73.40-75.27)
MLP	74.18 (73.61-74.75)	73.22 (69.89-76.55)	75.13 (72.17-78.08)
Logistic regression	74.39 (73.72-75.05)	75.96 (75.33-76.59)	72.83 (71.71-73.95)
Linear SVM	74.38 (73.77-74.98)	76.35 (75.82-76.87)	72.40 (71.30-73.50)
*k*NN	68.76 (68.33-69.20)	72.53 (72.13-72.92)	65.00 (64.24-65.76)
ESI	56.57	53.21	59.92

All our models significantly outperformed triage nurses in predicting hospitalization, as shown in Figure 5. Table 2, meanwhile, shows that every model we tested had significantly higher true positive and true negative rates than nurses. However, both achieved lower accuracy in predicting hospitalization compared to predicting ICU admission. One reason for this could be the datasets' size: the dataset we used to train these models was approximately seven times larger than the one we used to train the models used for ICU admission. As a result, the former set of models may have been prone to overfitting, which happens when a model tries to fit exactly against its training dataset and consequently performs poorly on new, previously unseen testing data. Meanwhile, another reason for our models' being better at predicting ICU admission is that ICU patients may have more anomalous vital signs, or tend to be older, making them easier to identify. This will be further explored in the next section, which discusses the influence of different predictor variables.

Feature importances

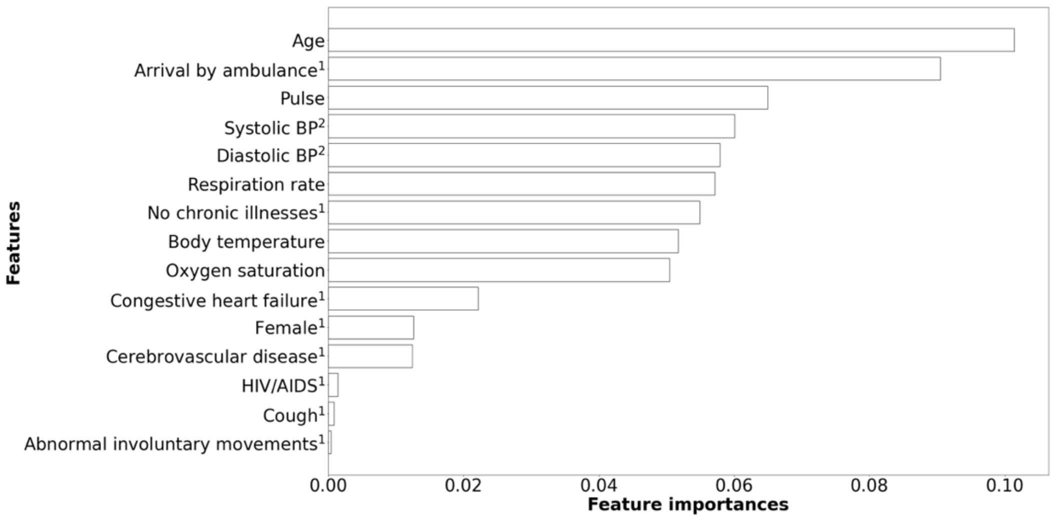


Figure 8. The 15 most important features, along with their importances, in predicting ICU admission.

[1]: yes/no categorical variables; [2]: BP = blood pressure

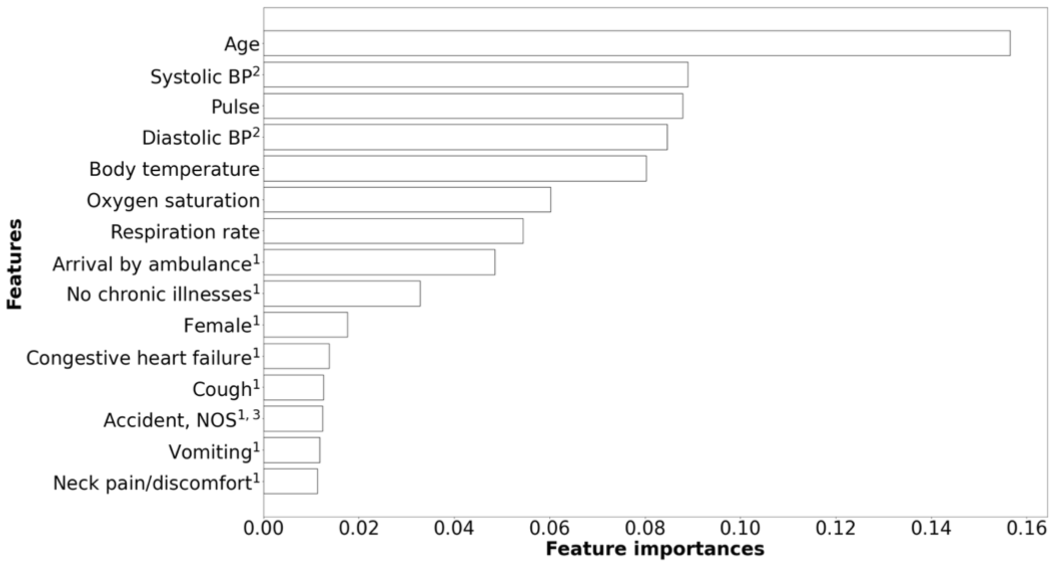


Figure 9. The 15 most important features, along with their importances, in predicting hospitalization.

[1]: yes/no categorical variables; [2]: BP = blood pressure; [3]: NOS - Not otherwise specified, which means there were enough symptoms for a patient to be diagnosed with an illness, but no diagnosis was made

After all the models were tested, the random forest model was used to calculate the feature importance of each variable in the dataset. Feature importance (on the x-axis in Figures 6 and 7), in the context of this study, refers to a variable's influence in predicting whether a patient will end up in the ICU (Figure 8) or hospitalized (Figure 9).

As seen in the figures, a patient's age, as well as vital signs such as pulse rate and blood pressure, were some of the most important features in both categories. Age is a very common deciding factor in the strength of an individual, with ages in the lower or higher extrema being more prone to injuries and diseases. Meanwhile, vital signs are the basic measurements of one's general health. Systolic blood pressure can also signify an individual's pain level. Unique to ICU admission is a patient's arrival by ambulance, a factor which is high on the list probably because only patients who would need critical care in the first place are brought in via ambulance. Meanwhile, the ranking of importance of body temperature is higher in hospitalization likely because while abnormally high body temperature is a cause for concern (and hospitalization), it alone is not enough to warrant something as dire as ICU admission.

Lastly, rounding out the 15 most important features were categorical variables such as congestive heart failure, cerebrovascular disease, or cough. The first two, along with HIV/AIDS, were the only chronic illnesses included in the dataset – they refer to a patient's medical history, rather than a recent diagnosis. However, it is important to note that patients with these diseases may be much more vulnerable than others, explaining their relatively high importance. Meanwhile, factors such as "cough" or "neck pain/discomfort" were likely good predictors of hospitalization because they are very common complaints – not necessarily because they make our models more accurate. While other complaints, such as "gastrointestinal bleeding", could naturally be interpreted by nurses as more serious, complaints like those rarely appeared in the dataset. Instead, since complaints such as "cough" appear in the dataset more frequently, our machine learning models had a larger sample size from which to assess their influence on a patient's disposition, which contributed to their relatively high importance.

Android Application

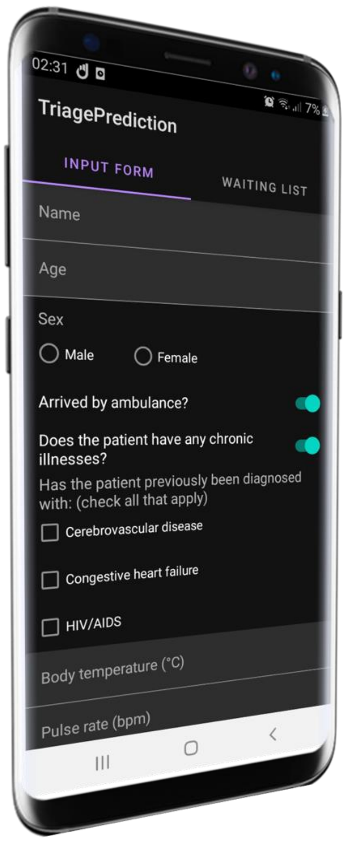


Figure 10. The application's first screen, containing its input form.

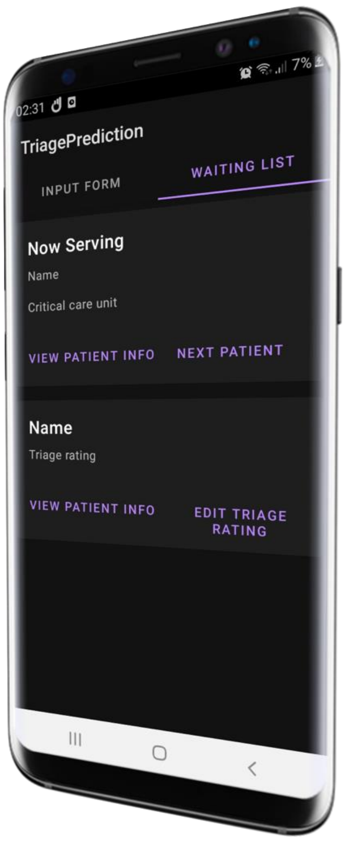


Figure 11. The application's waiting list, which sorts patients by priority.

Our Android application contains two screens. On the main screen, the user inputs a patient's age, sex, mode of arrival, vital signs, previously diagnosed chronic illnesses, and chief complaint (Figure 10). Upon clicking "submit", this data is sent to our machine learning models, which make predictions. Each patient's predicted triage classification is visible on the second screen, where the user may view the list of patients currently waiting in line for treatment (Figure 11).

SUMMARY AND CONCLUSION

This study met its major objective of creating an Android application that accepts patient data and uses it to predict their triage classification. Moreover, we met our specific objective of creating a model that has a significantly higher accuracy than nurses using the Emergency Severity Index in predicting both ICU admission and hospitalization. As such, our

Android application may offer significant advantages over conventional triage: its higher accuracy and ability to return predictions instantaneously could potentially help save lives. However, we do not intend for our application to replace nurses – instead, we intend for it to be used as a feedback tool to help them.

In future studies, additional models, or different configurations of the models we used, may be more effective. Future researchers may also use techniques such as natural language processing or chatbots to facilitate a more patient-oriented triage experience. Lastly, more data may be available in the future, which could allow the creation of more accurate models.

REFERENCES

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, *46*(3), 175. https://doi.org/10.2307/2685209

Castelvecchi, D. (2016). Can we open the black box of AI? *Nature*, *538*(7623), 20–23. https://doi.org/10.1038/538020a

Chu, X., Ilyas, I. F., Krishnan, S., & Wang, J. (2016). Data Cleaning: Overview and Emerging Challenges. *Proceedings of the 2016 International Conference on Management of Data*, 2201–2206. https://doi.org/10.1145/2882903.2912574

Elragby, O. (2019, March 1). The ultimate guide to data cleaning. *Towards Data Science*. https://towardsdatascience.com/the-ultimate-guide-to-data-cleaning-3969843991d4

Emergency Nurses' Association. (n.d.). *ESI*. Retrieved July 30, 2021, from https://www.ena.org/education/esi

Gilboy, N., Tanabe, P., Travers, D., & Rosenau, A. (2011). *Emergency Severity Index (ESI): A Triage Tool for Emergency Department Care* (Vol. 4). Agency for Healthcare Research and Quality. https://archive.ahrq.gov/professionals/systems/hospital/esi/esihandbk.pdf

Ho, T. K. (1995). Random decision forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*, *1*, 278–282. https://doi.org/10.1109/ICDAR.1995.598994

Hong, W. S., Haimovich, A. D., & Taylor, R. A. (2018). Predicting hospital admission at emergency department triage using machine learning. *PLOS ONE*, *13*(7), e0201016. https://doi.org/10.1371/journal.pone.0201016

Kwon, J., Lee, Y., Lee, Y., Lee, S., Park, H., & Park, J. (2018). Validation of deep-learning-based triage and acuity score using a large national dataset. *PLOS ONE*, *13*(10), e0205836. https://doi.org/10.1371/journal.pone.0205836

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Michie, D. (1968). “Memo” Functions and Machine Learning. *Nature*, *218*(5136), 19–22.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Ng, A. (2013). *Machine learning and AI via brain simulations*. https://ai.stanford.edu/~ang/slides/DeepLearning-Mar2013.pptx

Noble, W. S. (2006). What is a support vector machine? *Nature Biotechnology*, *24*(12), 1565–1567. https://doi.org/10.1038/nbt1206-1565

Pines, J. M., Pilgrim, R. L., Schneider, S. M., Siegel, B., & Viccellio, P. (2011). Practical implications of implementing emergency department crowding interventions: Summary of a moderated panel. *Academic Emergency Medicine*, *18*(12), 1278–1282. https://doi.org/10.1111/j.1553-2712.2011.01227.x

Raita, Y., Goto, T., Faridi, M. K., Brown, D. F. M., Camargo, C. A., & Hasegawa, K. (2019). Emergency department triage prediction of clinical outcomes using machine learning models. *Critical Care*, *23*(1), 64. https://doi.org/10.1186/s13054-019-2351-7

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., & Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354–359. https://doi.org/10.1038/nature24270

Sperandei, S. (2014). Understanding logistic regression analysis. *Biochemia Medica*, 12–18. https://doi.org/10.11613/BM.2014.003

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z.-H., Steinbach, M., Hand, D. J., & Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, *14*(1), 1–37. https://doi.org/10.1007/s10115-007-0114-2

APPENDICES

APPENDIX A: SAMPLE SOURCE CODE

```
#Code for training and testing machine learning models

X = dataset.iloc[:, :-2].values
y = dataset.iloc[:, -2].values #Dependent variable (to be predicted)
val = dataset.iloc[:,-2:]

val = val.dropna() #Get nurses’ predictions from dataset
ytrue = val.iloc[:,-2].values
ypred_esi = val.iloc[:,-1].values

under = RandomUnderSampler(sampling_strategy=1) #Undersample the dataset
X, y = under.fit_resample(X, y)
plt.figure(figsize=(16,5))

def evaluate(classifier, axes, graph, name):
    tprs = []
    aucs = []
    cv = KFold(n_splits=10, shuffle=True) #10-fold cross-validation
    f1scores = []
    for i, (train, test) in enumerate(cv.split(X, y)):
        classifier.fit(X[train], y[train]) #Feed the training datasets to
the model for training
        y_pred = classifier.predict(X[test]) #Test the model’s accuracy by
having it predict values in the testing dataset
        f1scores.append(f1_score(y[test], y_pred))
    plt.bar(name, np.mean(f1scores), color='#5a8cdb', label="{}:
{:.3f}".format(name, np.mean(f1scores)))


classifiers = [('Multilayer perceptron', MLPClassifier(alpha=0.1)),
('Random forest', RandomForestClassifier()),('Logistic regression',
LogisticRegression(max_iter=200,solver='liblinear',penalty='l1')),
('Gradient boosted trees', XGBClassifier()),
('Linear SVM', LinearSVC(dual=False)),
('k-nearest neighbors', KNeighborsClassifier(n_neighbors=180))]
#List of machine learning models to be tested
axes = plt.gca()

for name, classifier in classifiers:
    evaluate(classifier, axes, plt, name) #Train each model

f1esi = f1_score(ytrue, ypred_esi)
plt.bar("ESI", f1esi, label="ESI: {:.3f}".format(f1esi), color='orange')
axes.set_ylabel("$F_{1}$ scores")
legend=plt.legend()

plt.show() #Show bar graph containing each model’s F1 scores
```

APPENDIX B: SAMPLE SOURCE CODE (cont.)

```
#Java code for Patient class (Patient.java)

package com.example.triageprediction;

import android.widget.CheckBox;

import com.google.android.material.switchmaterial.SwitchMaterial;
import com.google.android.material.textfield.TextInputEditText;

import java.util.ArrayList;

public class Patient {

    public ArrayList<String> chronic;
    public ArrayList<String> rfvs;
    public String sex;
    public int prediction;
    public String name, age, temp, pulse, respr, sbp, dbp, popct, arrems,
nochron;

    Patient(String name, String age, String sex, String arrems, String
temp, String pulse, String respr, String sbp, String dbp, String popct,
String nochron, ArrayList<String> chronic, ArrayList<String> rfvs, int icu,
int hospitalized) {
        this.name = name;
        this.age = age;
        if (sex == "0.") this.sex = "Male";
        else this.sex = "Female";
        if (arrems == "0.") this.arrems = "No";
        else this.arrems = "Yes";
        this.temp = temp;
        this.pulse = pulse;
        this.respr = respr;
        this.sbp = sbp;
        this.dbp = dbp;
        this.popct = popct;
        if (nochron == "0.") this.nochron = "The patient has chronic
illnesses.";
        else this.nochron = "The patient has no chronic illnesses.";
        this.chronic = chronic;
        this.rfvs = rfvs;
        if (icu == 1) this.prediction = 1; //ICU patient = 1, other
hospitalization = 2, discharge = 3
        //We assigned numbers so the priority queue can sort patients
(minimum value = 1 – for ICU patients, who are at the head of the queue)
        else if (hospitalized == 1) this.prediction = 2;
        else if (icu == 0 && hospitalized == 0) this.prediction = 3;

    }

}
```

Java code for the application’s waiting list (WaitingList.java)

```
package com.example.triageprediction;

import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;
import static com.example.triageprediction.InputForm.queue;

public class WaitingList extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.waitinglist, container,
false);
        final TextView patientName = view.findViewById(R.id.patientname);
//Access text box containing patient name
        final TextView designation = view.findViewById(R.id.designation);
//Access text box containing patient designation

        Button button = view.findViewById(R.id.nextpatient);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (!queue.isEmpty()) {
                    Patient next = queue.poll(); //If the queue is not
empty, get the patient at the head of the queue
                    patientName.setText(next.name); //Put patient name on
screen
                    int p = next.prediction;
                    if (p == 1) designation.setText("Critical care unit –
ESI 1/2"); //Put patient designation on screen
                    else if (p == 2) designation.setText("Other
hospitalization – ESI 2/3");
                    else if (p == 3) designation.setText("Discharge – ESI
4/5");
                } else {
                    System.out.println("Queue is empty");
                }
            }
        });

        return view;
    }
}
```

APPENDIX D: DATASETS

The dataset used to train our machine learning models was built from 11 different datasets, one from each year from 2007 to 2018, from the U.S. National Center for Health Statistics (2018). Since each of these datasets contained approximately 30000 patients and more than 500 variables, there is no space to include them in this appendix.

All eleven datasets were, however, compiled into a single dataset with 182230 patients and 100 variables. This appendix will list down all 100 variables and include the data of the first five patients in the dataset.

The first nine variables in the dataset consisted of basic patient information and initial vital signs. Categorical variables are marked as either "0" or "1", with each of those numbers pertaining to a different category.

Table 1. The first nine variables in the dataset (in order): age, biological sex (0: male; 1: female), arrival by ambulance (0: no; 1: yes), body temperature in degrees Fahrenheit, pulse rate, respiration rate, systolic blood pressure, diastolic blood pressure, and pulse oximetry (oxygen saturation).

AGE	SEX	ARREMS	TEMPF	PULSE	RESPR	BPSYS	BPDIAS	POPCT
40	1	1	99.1	90	16	129	75	99
76	0	0	97.5	71	16	167	82	98
27	1	0	98.4	89	20	118	76	98
59	0	1	97.6	85	16	124	95	98
33	0	0	98.2	80	18	156	92	98

The next four variables are categorical variables, saying whether a patient had a particular chronic illness:

Table 2. The next four variables in the dataset. The first three refer to chronic illnesses: whether the patient has cerebrovascular disease, congestive heart failure, or HIV/AIDS (in that order). If the patient has none of these, then the fourth column, "NOCHRON", for "no chronic illnesses", is marked with a "1".

CEBVD	CHF	EDHIV	NOCHRON

0	0	0	1
0	0	0	1
0	0	0	1
0	0	0	1
0	0	0	1

The next four variables pertain to a patient's status in the emergency department. These were used to clean the data: if any patient was marked "1" in any of these categories, they were removed because they were not able to properly undergo triage.

Table 3. The next four variables in the dataset, which say whether a patient left before being seen, left against medical advice, was dead on arrival, or died in the emergency department.

LWBS	LEFTAMA	DOA	DIEDED
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

The remaining 83 variables in the dataset were one-hot encoded categorical variables containing patients' reasons for visiting the emergency department. Due to the size of this section of the dataset, only three of them will be displayed here:

Table 4. Three of the 83 reason for visit columns. A few more examples of these can be seen in Figures 14 and 15.

Chest pain	General weakness	Vomiting
0	0	0
0	0	0

0	0	0
0	0	0

【評語】190042

This project develops an Android APP for triage prediction in hospital emergency departments. The topic is interesting, timely, and very important for hospitals. The authors conducted a set of experiments and compared the accuracy performance of several ML methods in this work. Overall, this is very nice work and of some practical value. It would be great if the authors could provide more detailed information about the datasets used in this research, as well as collect feedback from nurses/doctors to further enhance the proposed work.