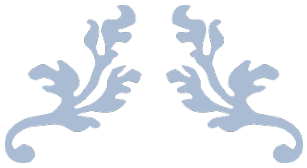

ICU Mortality Prediction Using Machine Learning Methods

Gregory Arbour
Department of Mathematics & Statistics, York University
MATH 6001: Survey Paper
Dr. Steven Wang
August 25, 2020.

Table of Contents

Introduction

The primary goal of this Survey Paper is to develop and evaluate a predictive model to predict mortality in a hospital ICU population. The research question and data were obtained from a past data science competition hosted by PhysioNet. Specifically, the "WiDS (Women in Data Science) Datathon 2020: ICU Mortality Prediction

[i].

PhysioNet is a platform that was established in 1999 in collaboration with the United States' National Institute of Health where one of the aims was to promote the use of data driven medical research and development of diagnostic or clinical treatment tools. It began as a cooperative project including "computer scientists, physicists, mathematicians, biomedical researchers, clinicians, and educators at Boston's Beth Israel Deaconess Medical Centre/Harvard Medical School and MIT". PhysioNet achieves this aim through its three complementary components. It provides:

1. An extensive archive of biomedical datasets, including time series, recording of physiologic signals such as heart rate, clinical imaging and other support data, taken from a wide array of studies and relating to physiologic areas such as cardiac death, epilepsy and gait disorders.
2. A library of software tools for processing and analyzing the available datasets
3. A collection of tutorials and educational materials that distills the accumulated knowledge of its past contributors

In addition, PhysioNet hosts numerous data science competitions per year that, given the high-quality data sets and compelling research questions, attract widespread attention and hundreds of participating teams. Past competitions have seen hundreds of entrants and dozens of papers published. Additionally, because the datasets are publicly available following the conclusion of the competition, many entrants also post their source code to public GitHub repositories so that others may access and reproduce the analysis.

The "WiDS Datathon 2020: ICU Mortality" competition hosted by PhysioNet in January of 2020 was a collaborative effort between the Global WiDS team, the West Big Data Innovation Hub[ii] and the WiDS Datathon Committee[iii] and used data provided by MIT's Global Open Source Severity of Illness Score[iv]. The Kaggle platform was used to track submissions and provide a message board for competition entrants.

The objective of the competition is to create a model that uses data from the first 24 hours of a patient's admission to an intensive care unit to predict the patient's survival. The dataset included more

[i] https://physionet.org/content/widsdatathon2020/1.0.0/
[ii] https://westbigdatahub.org/
[iii] https://www.widsconference.org/committee-2020.html
[iv] https://gossis.mit.edu/

than 130,000 hospital visits spanning a one-year timeframe. The metric used to evaluate model performance is Area under the Receiver Operator Curve (AUC) on a held-out test dataset. The test data set, excluding the labels for the outcome variable (survival or death) was provided to competition entrants. This Survey Paper used the same objective although explored several metrics for evaluating model performance in addition to AUC.

Literature Review

The application of machine learning methods to the healthcare industry is an extremely active area of research. Healthcare processes are often complex as well as data rich, which lends itself well to the use of machine learning. Intensive Care Units (ICUs) in particular have a patient population that is on average much more acutely ill and therefore more closely monitored. Different machines are used for ongoing monitoring of various vital signs and a greater number of laboratory tests are ordered to investigate or monitor different physiological markers. This results in a large quantity of data being generated for each patient admitted to an ICU.

Current machine learning research spans a huge variety of topics from a diverse array of patient populations, geographies and research questions. Machine learning has been used to predict a variety of clinical outcomes, including onset of sepsis[34], readmission to the unit or hospital[35], kidney failure, patient mortality, and many others. This literature review will focus specifically on the application of machine learning algorithms applied to predicting mortality in hospital intensive care units.

Practical Applications

As the studies included in this literature review are taken largely from real hospitals using actual patient data, it begs the question as to how the results can be best applied in the real world or incorporated into a hospital's workflow. The most obvious application is for the model to supplement the intensivist's decision-making process by highlighting which patients are at higher risk of death[1]. The sooner and more accurately this determination can be made, the better chance there is to modify the patient's course of treatment and affect a more positive outcome. Given the large number of clinical signals in intensive care units, a low false positive rate is critical to avoid alarm fatigue. At the same time, the predictive model must be sensitive enough to identify patients before the intensivist would themselves have been able to make this determination. A model that generates a high number of false positives or has lower sensitivity than currently exhibited by humans is impractical for real world deployment.

Another application is to use mortality predictions to enable the hospital to set improvement targets for quality improvement efforts. For example, a study by Delahanty, Kaufman, & Jones (2018) included 53 hospitals in a single health care system. The implementation of a prediction model allowed for between hospital comparison and the establishment of performance benchmarks.

Lastly, machine learning models that output a probability of survival allow for better communication between care providers and the patient or family members[19].

Research questions included

Patient mortality refers to whether or not the patient dies during some pre-specified time frame. It is a binary outcome, where the only two possible values are 'death' or 'survival'. The most commonly used time frame used is the remainder of the patients' hospital stay. This makes sense as the primary goal of the healthcare system should be to prevent patients from dying while under their care. Less common is a 30-day or even one-year time frame (from the time of discharge from hospital).

Sub-populations available for study

Even within the critical care population there are many sub-groups that have been sectioned out for more focused mortality prediction. These include children (under the age of 18)[6,21], COPD patients[20], patients recovering from cardiac surgery[2], influenza patients[25], patients who are being treated for cardiogenic or septic shock[23], post-surgical patients[17] and many others.

Many of the databases that are used for analysis are proprietary and not open for public use. However, a number of prominent publicly accessible databases are featured in the research, particularly the MIMIC-II (Medical Information Mart for Intensive Care) and MIMIC-III databases. These are large, freely available databases that include more than 40,000 detailed patient records gathered from the Beth Israel Deaconess Medical Center between 2001 and 2012[36]. The records have been deidentified of personal information. Though some credentialing is required to gain access the barriers are much lower than for other databases. This freedom of access as well as the detailed nature of the data collected means it is one of the most highly studied ICU patient populations in the world and was used in a large portion of the studies in this literature review.

Mortality rate

One of the important considerations in building a model to predict patient mortality is that the outcome variable is usually highly imbalanced. That is, the proportion of patients who die is much lower than those who survive. Where no special filtering was done, the population was comprised of all patients admitted to a hospital's adult ICU and the corresponding mortality rate was close to 10%[3, 4, 21]. However, when sub-populations were used instead this varied substantially, depending on the acuity of the sub-population. ICU patients who experienced unplanned extubations (removal of an endotracheal tube, meant to support the patient's ability to breathe) had a mortality rate of 17.6%[18]. The mortality rate was as low as 6% for a cohort of post-operative cardiothoracic surgery patients[2] and as high as 45.6% for patients who had been resuscitated following a cardiac arrest[19]. Even within the studies using the MIMIC-II or MIMIC-III databases there was significant variation in the mortality rate. This is explained by different sub-

populations within the database being used. For instance, MIMIC-III patients who had central lines had a mortality of 16.2%[5] where patients with acute kidney injury (AKI) were as high as 43.4%[8].

Sample size

Since machine learning methods are intended to be used with a large sample of data and, preferably, a large number of features, the vast majority of the reviewed studies fulfil these criteria. A typical study included 20,000-40,000 records. Only three studies had 100,000 records or more[12,28,31] and the largest had 217,289[28]. Many studies began with a large database like MIMIC-III and later settled on a much smaller subset of this once exclusion criteria were applied. For instance, a study by Jain, S., Sarkar, I., Stey, P., Anand, R., Biron, D., & Chen, E. (2018) examining COPD patients in the MIMIC-III database had 1,198 records. A small number of studies focused on even narrower populations and had corresponding small sample sizes. Hu CA, Chen CM, Fang YC, et al (2020) examined influenza patients following an epidemic in Taiwan between October 2015 and March 2016 and used a mere 336 patient encounters as their dataset.

Types of variables included

The studies varied in which particular features were available in their respective datasets or were selected for final inclusion in the model. Typically, these included some combination of demographic, physiological (including primary diagnosis and comorbidities), vital signs and lab values. Though less common, medications and ordered procedures were sometimes included[21].

The reasons for exclusion of certain variables were diverse. In some cases, the data simply wasn't available or would have been too expensive to collect. Meyer, Zverinski, Pfahringer, et al. (2018) deliberately excluded variables that they felt contained "traces of human intelligence". The rationale is that to properly compare whether a machine learning approach is more effective than clinical judgement it is important to strip out any lingering reflections of clinical judgement from the data used to train the model. Meyer et al. claimed, "This ensures that the models are exclusively trained on systemic patient properties rather than people's reactions to them". The implication for that particular research paper was that no information on specific tests that were ordered, such as blood transfusions or medications, were included in the dataset.

Many studies sought to produce a generalizable model that could potentially be deployed, after retraining on the local data, to other ICU's. These models therefore used only variables normally collected and commonly available in most modern ICU's. One study was designed specifically to be compared with the Simplified Acute Physiology Score (SAPS) for mortality and used the same variables in their model as are used to calculate SAPS[28].

The most common scenario was the researchers sought to maximize the predictive power of their model and therefore utilized all the data that was available to them. This is more in line with the ethos of the machine learning community, which tends to prioritize performance over other considerations like interpretability or generalizability.

Time horizon of included variables

Not only did the time horizon for mortality vary (death during hospital stay, within 30 days or within one year) but there was also a degree of variation among how long and what type of prediction window is used. Prediction windows fall into one of two categories: dynamic and static. Under a static prediction window all the data from a certain period, say the first 48 hours of the patient's admission to the ICU, are collected and used in the model to output a single number such as the probability of death within the next 30 days.

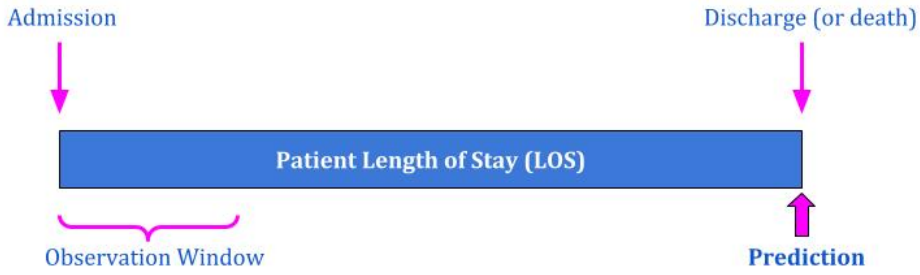

Figure 1: Static Prediction Window

A 48-hour static window was the most common time period used. Less common was the first 24 hours. One study sought to predict mortality at the outset of the admission to the ICU and another used data collected only in the first hour. The practical implications of these models are clearly different than those that gather a history of data on a patient for 24 or 48 hours. The former likely is designed to aid in triaging or initial care planning whereas the latter is much more useful for alerting the intensivist to potential negative outcomes that are likely to arise. Mayampurath et al. (2019) noted that "This timeframe [first 48 hours] was chosen in order to provide an early time point in a patient's admission to make predictions, while also allowing time for determining if a patient is responding to therapy."

Dynamic prediction windows, as the name suggests, use a rolling time period as opposed to a single, static time period. A series of predictions are made as new information about the patient is created and collected. For example, Kim, S., Kim, S., Cho, J., Kim, Y., Sol, I. (2019) produced a model which predicted the probability of death within the next 24 hours and made this prediction at 6, 12, 24, 48 & 60 hours after the patient's admission to the ICU. Each patient encounter has up to five different predictions

calculated (fewer in the case of the patient dying less than 60 hours in their admission). Raj, R., Luostarinen, T., Pursiainen, E., Posti, J., Takala, R., et al. (2019) used a dynamic prediction window every 8 hours (8, 16, 24 up to 120 hours).

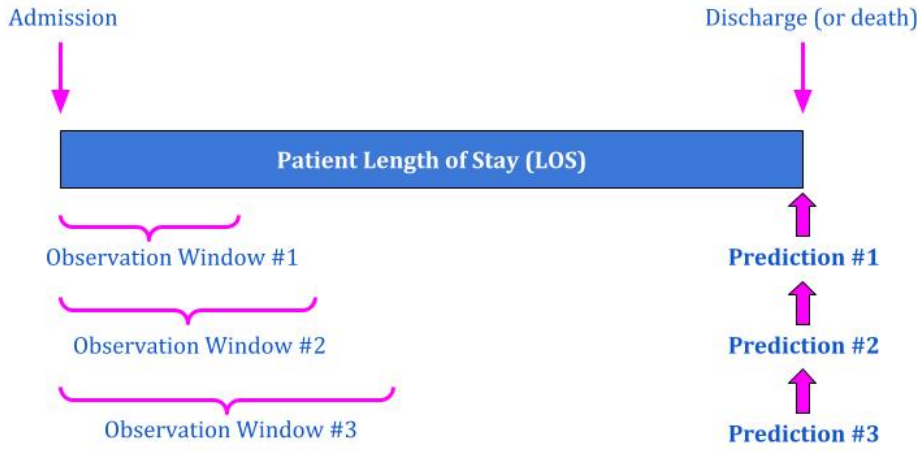

Figure 2: Dynamic Prediction Window

Under the dynamic prediction arrangement, the model generates a set of predictions for each individual patient encounter. Unlike a simple binary prediction model, evaluation of the dynamic prediction models is not as straightforward. Dynamic prediction windows also allow for the use of different machine learning models, unseen in other static prediction tasks. In particular, RNN or LTSM models are used[16].

Though much harder to deploy in a real-world environment, dynamic prediction systems likely provide the patient care team much more utility than a single, static prediction using data from a fixed time period. In the case where a patient has spent seven days in the ICU, it is of little bearing what the mortality prediction was five days prior.

Feature engineering

Cleaning data and performing feature extraction is typically the most time intensive portion of building a machine learning model (or indeed, any predictive model that uses large, complex data). Though not always explained in great detail, the two most common steps taken included one-hot encoding categorical variables and processing sequential (i.e. time series) continuous variables to extract a smaller number of features that still contained many of the most important information. Features such as first, last, min, max or average values as well as slope (rate of change) from the distribution of collected values over a period of time. The alternative to summarizing time series data in this manner is to maintain it in a sequential format, for instance one measurement every hour. In some cases, this still may require processing, for example, if a measurement is instead taken every 10 minutes. Different logic can be applied

to coerce this to an hourly measurement, say by taking the average of the six values over the hour, of the mode, min or max value that occurred within the given hour.

In the case of unstructured text data, different techniques were used to create useful predictor variables. Marafino et al. (2018) used term frequency-inverse document frequency (tf-idf) to identify the top 1000 most important words in the corpus of text. The tf-idf weight indicates how important a particular word is to a particular document in a collection of many documents. The weight of the word is increased if it appears many times in that document but decreased if it appears many times in the collection of other documents.

Weissman, G., Hubbard, R., Ungar, L., Harhay, M., Greene, C et al. (2018) created a variable for every possible 1-gram, 2-gram and 3-gram that occurred in the corpus of text and then used penalized logistic regression to select the top 500 (out of a total of 5,790) n-grams that were predictive of mortality. The 500 n-grams were then used as predictor variables in the other machine learning models that were constructed.

Clinical severity scores such as the Charlson Comorbidity Index or Diabetic Severity Index were sometimes already available as routinely collected variables in the dataset but in a few cases, they were calculated explicitly[15]. Aperstein, Y., Cohen, L., Bendavid, I., Cohen, J., Grozovsky, E. et al. (2019) also employed a custom tool for scoring gastrointestinal health. Though it is not a fully validated tool as of this writing, its use in their predictive model serves a similar purpose.

Missing Value imputation

The methods used to handle missing data were surprisingly naive. The most common method was mean/mode imputation or, for sequential data, either Last-Observation-Carried-Forward (LOCF) or Last-Observation-Carried-Backwards (LOCB). Usually a threshold of missingness was set and the record removed completed if, for example, 30% of the features in that record had a missing value[11]. A handful of studies utilized more sophisticated methods such as K-Nearest Neighbors imputation[12] or Multiple Imputation by Chained Equations (MICE)[3]. In one study, an autoencoder imputation method was attempted but did not show to be an improvement over mean imputation[28], which was then used instead on the grounds of preferring simpler solutions over more complex ones.

Outlier handling

Only a handful of studies made explicit mention of techniques used to handle outliers in the data. Kim, S et al. (2019) used clinical knowledge and experience to define a range of physiologically possible values and remove values outside this range. Meiring, C., Dixit, A., Harris, S., MacCallum, N., Brealey, D. et al. (2018) used the Ye-Johnson power transformation on variables that contained possible outliers to

minimize their impact. Another approach was to apply a heuristic such as removing data points that are greater or less than +/- 5 standard deviations from the mean.

Class imbalance problem

The outcome variable, mortality, was in most cases imbalanced between positive and negative cases. This is problematic because many binary classifiers are designed in such a way to favour prediction of the majority class (since the objective is usually to maximize accuracy, as opposed to some balance of sensitivity and specificity). Therefore, most studies had to consider the issue of handling the class imbalance problem. Several options exist for accomplishing this.

The first is to simply ignore the problem and assume that the use of random sampling to create the train and test sets will be sufficient. This was the approach used in the majority of studies. Stratified sampling, so that the train and test sets have roughly the same proportion of cases with a positive outcome, is a more direct way of reliably achieving this, particularly if the sample size is small. However, both of these methods allow the imbalance to persist in the dataset used to train the model.

Xu Z, Luo Y, Adekkanattu P, et al. (2019) reconstructed the training set using case control matching. For each record in the minority class (death) a corresponding case from the majority class (survival) was chosen based on age, gender, and similar APACHE II and Charlson Morbidity index scores. The resampled training set was therefore balanced between the two levels of the outcome variable.

Finally, oversampling of the minority class[9] or undersampling of the majority class[2] was occasionally used. The first approach has the disadvantage of creating a bias where none exists and the second may be problematic if the number of records is small.

Resampling methods

The goal of a predictive model is not to predict data that has already been observed, but instead to make predictions on as of yet unseen data. This is approximated by first splitting the dataset into training and testing data sets. The model is trained on a training data set and then evaluated based on its performance on the hold out test dataset. There is no hard and fast rule for what proportion of the data should be reserved for the test set[41] as, like many other things, it depends on the specific context. The goal is to have both a large enough set to properly train the model while still having enough data reserved for an unbiased evaluation of model performance afterwards. The larger the data set the less the choice of proportion matters. With small data sets (<1000 records) the chosen proportion can have significant implications. If the training set is too small, the model may not fit well and will exhibit high bias. On the other hand, if the test set is small it may not be an accurate reflection of the performance on the fully trained model. Studies used a training set containing between 66-90% of the original dataset.

Model parameters are usually derived through k-fold cross validation on the training set. Nearly every study used k-fold cross validation with either 5 or 10 folds. Meiring, C. et al. (2018) used a leave-group-out approach, forgoing the creation of a test set altogether. A few studies also sought to further validate their models using external data sources[13]. Xu Z. et al. (2019) postulated that different models would need to be constructed for different stages of the disease. Rather than train a single model on the entire training set, the data was first split into three subsets for each stage in the disease progression, and then a unique model was trained on each subset.

Comparison to baseline models

To justify the use of a complex, computationally intensive approach like machine learning, it behooves the researcher to demonstrate that it is superior to some baseline method. The traditional approach to a binary outcome such as mortality is logistic regression, though this is fairly limited when there are complex or nonlinear relationships involved. Several clinical prediction tools have been developed and validated that serve a similar purpose but with a higher degree of accuracy. The Acute Physiology and Chronic Health Evaluation (APACHE) II score, Simplified Acute Physiology Score (SAPS) II and Sequential Organ Failure Assessment (SOFA) score are three such measures. However, it is important to note that these scores are meant to provide the average mortality of a group of patients falling into a certain category, not predict an individual's probability of survival[37].

A number of studies have been undertaken to test the predictive power of these tools in real life clinical settings. The primary outcome measure is often the area under the curve (AUC) of the receiver operating characteristic (ROC) curve. Choi, J., Jang, J., Lim, Y., Jang, J., Lee, G. et al. (2018) found AUC values for the three severity scores between 0.72-0.77 when applied to a cohort of post-cardiac arrest patients who were treated with therapeutic hypothermia. Falcão, A., Barros, A., Bezerra, A. et al. (2019) studied a group of 3,568 patients in a post-surgical ICU and saw AUC values ranging from (0.74-0.79). Many other studies have been conducted on various sub-populations and with mixed results. It would seem that the tools perform quite well for some patient populations and poorly for others. In any event, it is clear that they provide an important competing indicator that should be taken into consideration when predicting mortality in an ICU scenario.

While APACHE, SAPS & SOFA were the most widely used comparison measures[2, 3, 13], a small number of others measures were also used. Awad, A., Bader-El-Den, M., McNicholas, J., & Briggs, J. (2017) considered the National Early Warning Score (NEWS). Moll, M., Qiao, D., Regan, E. et al. (2020) compared their results to those generated by the body-mass index (BMI), obstruction, dyspnea and exercise capacity (BODE) index, Chiew, Liu, Wong, Sim, Abdullah (2019) compared to the Combined Assessment

of Risk Encountered in Surgery (CARES) and Kang, M., Kim, J., Kim, D., Oh, K., Joo K. et al. (2018) used MOSAIC, an abbreviated prediction model specifically for patients with acute kidney injury. Two studies used a standard logistic regression model as a baseline comparison[22, 25].

A more important baseline comparison is predictions generated by the intensivist overseeing the patient's care. For a number of reasons, it is often impractical to expect the physician to be able to dedicate sufficient time and energy into making this assessment in a clinical setting. The deployment of automated tools should, ideally, fill in this gap if proven to be a more reliable tool in comparison with clinical judgement. Unfortunately, due either to infeasibility or the monetary expense of gathering the physician generated predictions, only a single study by Ruiz, V. et al. (2019) in the literature review took this approach.

In nearly all cases the machine learning approach was superior to the baseline comparison method. Possibly this is due to some kind of publication bias, since there is little demand for novel machine learning models that do not outperform existing, simpler methods.

Models used

Approximately 18 different machine learning model types were used.

Table 1: Count of models used in literature review

Algorithm	**n**
Random Forest	18
Penalized Logistic Regression	13
Neural Network	11
Support Vector Machine	10
Gradient Boosted Machine (usually XGBoost)	10
Recurrent Neural Net	3
AdaBoost	3
Naive Bayes	3
Convolutional Neural Net	2
Long Short-term Memory Network (LSTM)	2
Fuzzy Rule Based Classifier	2
Decision Tree	1
K-Nearest Neighbors	1
Projective Adaptive Resonance Theory (PART)	1
Classification and Regression Tree (CART)	1
Multivariate adaptive regression spline (MARS)	1
Lazy K-star	1
BayesNet	1

Random Forests was the most popular model and often the best performing, demonstrating a versatility in the model as it was applied with success across many different subpopulations and predictor variable subsets. Penalized Logistic regression (either Ridge, LASSO or Elastic) was the second most popular, likely to do with its reasonably high predictive power coupled with ease and speed of training. Gradient Boosting Machines (GBM's) were used in ten different studies. Though not always explicitly stated, it can be assumed that this usually entailed the XGBoost variant. GBM's have only become popular within the past few years, which may explain their relatively rare use in the literature. The top five algorithms comprised 74% of the total (62/84) which is unsurprising considering they are the five most common binary classifiers used in modern machine learning.

Programming language Used

Where stated, it appears that the two most common programming languages used for analysis were R[3,4,18,22] or Python[8,12,19,21]. Scikit-learn was the most frequently mentioned Python library[8,21]and the Caret

library was most frequently mentioned in R[3,18]. Julia was used in two studies[14, 20]. Simple statistical analysis was often done in a different program before completing the remaining machine learning component in either R or Python. Raj, R. et al. (2019) used SPSS for this purpose.

Model interpretability

As machine learning methods become more widespread, model interpretability becomes a significant concern. Ahmad, M., Eckert, C., & Teredesai, A. (2018) noted that "interpretable [machine learning] thus allows the end user to interrogate, understand, debug and even improve the machine learning system."

In the context of healthcare, where important, high-stake decisions can be made based on the output of such a model, this has several implications. First, by allowing the end user to understand the model output it can aid in debugging or improving the model. In some cases, this might involve uncovering a bias or discriminatory behaviour[40] that was learned by the model. Since machine learning models can sometimes fail in spectacular (and embarrassing) ways, providing for some level of interpretability allows the human operator to work in tandem with the machine and better identify these occurrences. Second, it gives the human operator an opportunity to grow their own knowledge and understanding of the process. Lastly, it fosters trust in the model as clinicians are justifiably not quick to trust a tool that is unvalidated or entirely opaque in its decision-making process.

Only a few studies make a point to explore this issue. For many machine learning methods (deep learning especially) it is assumed to operate as a sort of 'black box' where interpretability is sacrificed in order to obtain better performance. Some algorithms, such as Random Forest or XGBoost, do facilitate extraction of some measure of feature importance though the result is still not as meaningful as a classical statistical model such as logistic regression. As the primary research question is related to predicting mortality, the natural progression is to ask, 'what causes that occurrence?' Though the many complex, nonlinear relationships are unlikely to be able to be captured and distilled in a meaningful way, highlighting which variables are most important to the model does provide some semblance of an answer. For example, Kim S. et al. (2019) used a built-in function to examine feature importance of the Random Forest model and Xu Z. et al. (2019) for the XGBoost model. SHAP (SHapely Additive exPlanations) is another library that works with XGBoost and was used by Hu et al. (2020) to visualize feature importance for the model.

Nanayakkara, S., Fogarty, S., Tremeer et al. (2018) used a LIME Explainer on a subset of records which the model predicted to have either a very high or very low mortality. The top ten features for each case and then examined. This gives some insight into what the model thinks are the most important variables for cases it feels very confident about, although more ambiguous cases are not explored in the same manner.

Ge, W., Huh, J., Park, Y et al. (2018) has a detailed list of feature importance weights and a corresponding discussion of their implications. These were derived by simply extracting the weights used for each feature in the model. The authors compiled a list for both features associated with mortality and features associated with survival. While vital signs (each comprises 48 separate features as they are sequential variables) were not in the top 20 of mortality, three vital sign features were in the top 20 predictors of survival. This indicates that a stable patient will usually have normal vitals, whereas an unstable patient may not always have abnormal vitals.

Conclusion

Though most papers used similar outcome metrics (namely AUC, and perhaps a few other complementary measures) the diverse nature of which data was allowed into the study and the temporal windows used for the training data (e.g. the first two days of the patient's admission to the ICU) as well as the mortality window (e.g. 30-day survival) means direct comparisons between studies is not always possible. This is true even when patient populations are similar or even identical, as is the case for the many studies using the MIMIC-III database.

It is difficult to synthesize a set of conclusions that hold true across all 33 studies included in the review, especially given the high degree of variance in patient population, sample size, predictor variables, methods used and even the research question that was being answered. However, several trends did emerge, even if they weren't true in every case.

In keeping with standard practices for training machine learning models, k-fold cross validation was nearly always used. In some cases, a different resampling method was used, though this was rare.

There was often some attempt made at explaining variable importance of the model. This is straightforward in the case of logistic regression and gets progressively harder with the complexity of the model used. Either out of the box tools we used, like those with Random Forest of GBM's implementations in R and Python, or in rarer cases an additional tool like a LIME Explainer was used. Model explainability in all cases appeared to be of a lower priority that model performance.

Finally, Machine Learning methods tended to outperform simpler methods such as severity scores (APACHE, SAPS) and logistic regression. More sophisticated machine learning methods like Random Forests or Neural Networks tended to outperform simpler ones like Naive Bayes or Decision Trees. This is not necessarily a triumph for machine learning over classical statistical methods as there is almost surely a strong selection bias at play. Since the premise of each study is to utilize machine learning, 'file drawer bias' implies that negative results where the machine learning model did not outperform a classic method in some way would be far less likely to be published.

Description of Dataset

The dataset contains 91,713 complete records and 186 variables of patients admitted to an ICU in one of 66 different hospitals. It also included 39,312 incomplete records that comprised the test dataset (incomplete only in the sense that they were missing values for the outcome variable). To register a final score in the competition, participants output a set of predictions on the test dataset and computer script computes and outputs an AUC. For the purposes of this analysis, the 39,312 records in the test dataset were discarded and the training dataset, which contains a complete set of labels, was split into a new train-test dataset so that the models could be evaluated without needing to submit through the PhysioNet competition platform.

The data covers the first 24 hours of their stay in the ICU. The primary outcome variable is "mortality", which is a binary variable that only takes on values of 1, indicating death during the hospital stay, or 0, indicating survival. The inclusion criteria is that the patient must have been admitted to the ICU during their hospitalization, although the mortality variable tracks whether or not the patient died in the hospital. This could mean the patient died while in the ICU *or* in a different unit after being transferred from the ICU, but still remaining in hospital. The outcome variable is imbalanced as 90% of patients survive their hospital admission.

The variables are a mix of binary, categorical and continuous data types. Though largely composed of values for laboratory tests or physiological measurements, the dataset also included a number of demographic variables as well as other ICU specific measures, such as the widely used Glasgow Coma Scale (GCS).

Table 2: Overview of predictor variables in dataset

Type of Variable	**Number of Variables**	**Example(s)**
Demographic	16	Age, weight, ethnicity
APACHE (ICU specific variables)	40	Primary Diagnosis, Glasgow Coma Scale
Vitals	52	Blood Pressure, Heart Rate, Peripheral Oxygen saturations
Lab values	76	Albumin concentration, creatinine concentration, proportion of red blood cells

The dataset is in many ways typical of what one might expect from a hospital. It contains standard demographic data like age, gender, BMI, ethnicity and admission source (e.g. Emergency Department). It also contains the most medically relevant variables such as diagnosis and comorbidities like hypertension or diabetes. A number of ICU-specific variables, such as different components of the Glasgow Coma Scale or the body system (e.g. respiratory, cardiovascular) of the admission diagnosis are also available. What makes this dataset suitable for a machine learning approach is the large number of physiologic measurements and lab results, such as diastolic or systolic blood pressure, lactate concentration, bilirubin concentration, and many others. Human physicians are able to marshal their vast knowledge of modern medicine and human biology to make determinations involving a large number of variables. It is only in certain cases where a high bilirubin concentration matters. The motivation for using a machine learning model is to uncover these complex, nonlinear relationships and interdependencies.

The lab and vital sign variables were always given as a pair of the 'highest value in the first 24 hours' and the 'lowest value in the first 24 hours.' For example, the variable "highest bilirubin concentration of the patient in their serum or plasma during the first 24 hours of their unit stay" would be paired with the variable "lowest bilirubin concentration of the patient in their serum or plasma during the first 24 hours of their unit stay".

Given its importance to understanding a patient's health status, there were a large number of variables related specifically to blood pressure. There are several types of blood pressure (systolic, diastolic, mean) and several ways to measure a patient's blood pressure (invasively or non-invasively). Minimum and maximum values of the above permutations (e.g. Mean Blood pressure, invasively measured) were obtained for both the first hour of the patient's admission and the first 24 hours of the patients admission. There were 36 blood pressure variables contained in the dataset. Many of these were highly correlated with one another.

Binary variables were primarily related to the presence (or absence) of a specific comorbidity or other important feature. For example, "has_leukemia" indicates if the patient has been diagnosed with leukemia or "elective_surgery" indicates whether the patient was admitted to the hospital for an elective surgical operation.

Two variables of particular interest are "apache_4a_hospital_death_prob" and "apache_4a_icu_death_prob". These represent the probabilistic prediction of in-hospital mortality and in-ICU mortality, respectively, as calculated using the APACHE IVa guide. The prediction is calculated using an algorithm that was developed and validated to support clinicians to better predict mortality. It utilizes a set of lab values, Glasgow Coma Score values and some demographic data. These two variables provide a baseline prediction which allow for several additional queries into the proposed machine learning models. The models can be trained both with or without the two variables. In the latter case, we can examine if a

machine learning model is superior to the APACHE algorithmic prediction and in the former we can examine how much the machine learning model is improved by inclusion of the two variables.

A complete data dictionary is contained in the Appendix.

Data Exploration

Train/Test Data Partition

As part of the standard predictive modeling work flow, the first step before anything else was to split the data into separate training and testing sets. The split used was 80% in the training set and the remaining 20% in the test set. A simple random sample was used, as opposed to say, a stratified sample on the outcome variable. Given the large sample size of the dataset, a stratified sample was unnecessary. Checking the proportion of the outcome variable in the train and test sets after the split verified that the proportion of 90% survival was maintained in both data sets.

Though the same feature engineering and data pre-processing steps (for example, one-hot encoding a categorical variable) was performed on both the train and the test sets, the test set was otherwise completely unexamined until the final step of inputting the test sets into the models to evaluate the predictive power. All of the following data exploration was performed solely on the training set.

Table 1

The first step in any data analysis is to explore the dataset and learn about the scale, shape, missingness and relationship between the various predictor variables. A summary table was created for a quick reference to better understand categorical, binary and numeric variables. The complete table is too large to include in this report, so a representative sample of a few variables is shown below instead.

Table 3: Sample selection of Table 1

	0 (N=67038)	1 (N=6332)	Overall (N=73370)
factor(gender)			
F	30809 (46.0%)	2998 (47.3%)	33807 (46.1%)
M	36216 (54.0%)	3328 (52.6%)	39544 (53.9%)
Missing	13 (0.0%)	6 (0.1%)	19 (0.0%)
factor(ethnicity)			
African American	7055 (10.5%)	591 (9.3%)	7646 (10.4%)
Asian	831 (1.2%)	80 (1.3%)	911 (1.2%)
Caucasian	51565 (76.9%)	4944 (78.1%)	56509 (77.0%)
Hispanic	2756 (4.1%)	292 (4.6%)	3048 (4.2%)
Native American	555 (0.8%)	59 (0.9%)	614 (0.8%)
Other/Unknown	3238 (4.8%)	281 (4.4%)	3519 (4.8%)
Missing	1038 (1.5%)	85 (1.3%)	1123 (1.5%)
age			
Mean (SD)	61.8 (16.9)	68.6 (14.3)	62.3 (16.8)
Median [Min, Max]	64.0 [16.0, 89.0]	71.0 [16.0, 89.0]	65.0 [16.0, 89.0]
Missing	2915 (4.3%)	529 (8.4%)	3444 (4.7%)
sodium_apache			
Mean (SD)	138 (5.10)	138 (6.79)	138 (5.28)
Median [Min, Max]	138 [117, 158]	138 [117, 158]	138 [117, 158]
Missing	13746 (20.5%)	1099 (17.4%)	14845 (20.2%)

Plotting individual variables

To develop a better understanding of the behaviour of the numeric variables, each was plotted using a histogram or density plot. These plots were also used later to help identify possible outliers. Many variables, like age, were heavily skewed. The population of people admitted to hospitals, and ICU's in particular, tend to be older, so this is not surprising.

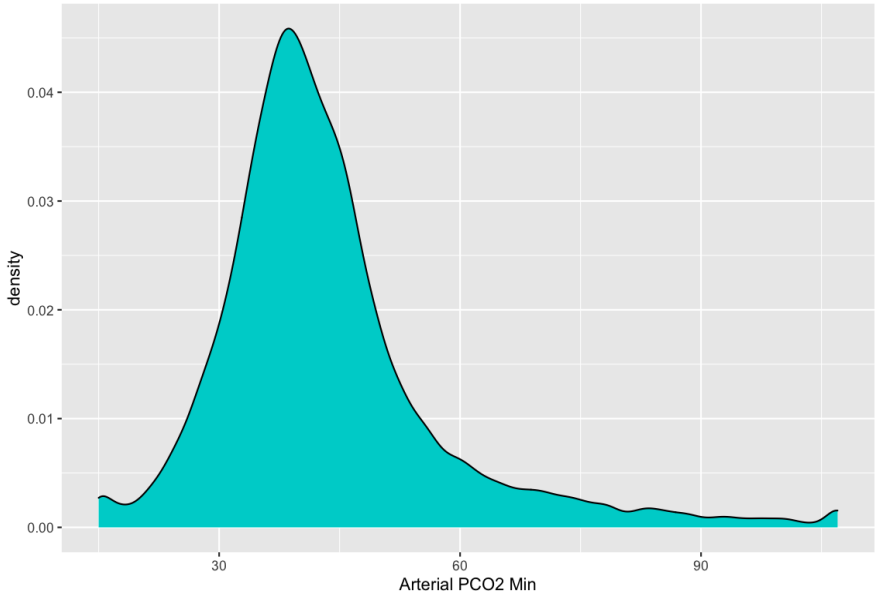

Figure 3: Density Plot of Arterial PCO2 Min

Many of the lab values were more normally distributed in appearance, also several exhibited large tails. These outlier patients on the extreme end of the distribution are of particular interest especially if these extreme values correlation with the outcome variable of in-hospital mortality.

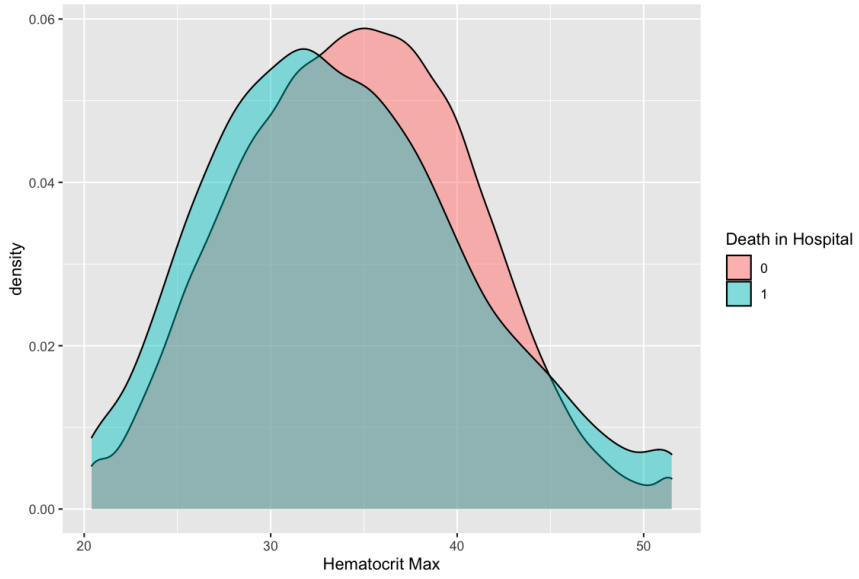

Figure 4: Density Plot Comparing Hematocrit Max across two factor levels

It is usually advisable in classification problems to plot individual predictor variables against the outcome variable, in this case “Death in Hospital” i.e. “in-hospital mortality”. In this way we can gain some insight as to the predictive power of that variable.

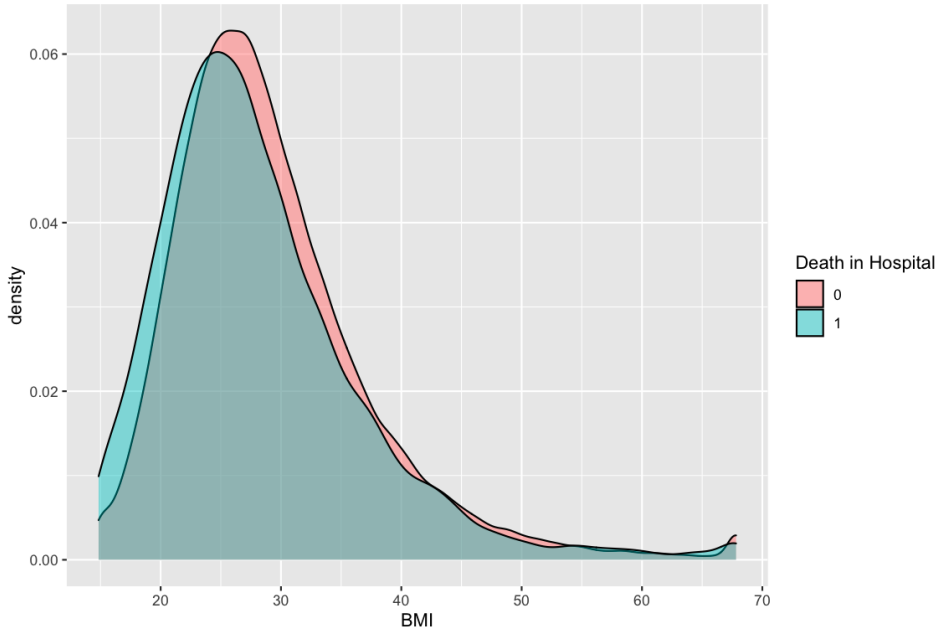

Figure 5: BMI vs In-hospital Mortality

Some variables display very little separation between the two classes of the outcome variable. For example, it would appear that patients are equally likely to survive across the entire range of possible BMI values. There is a slightly larger portion of patients at the lower end of the BMI scale who die within hospital, indicating that perhaps patients who are underweight have a lesser chance of surviving the admission. However, the differences are very slight and are unlikely to yield significant discrimination in the model.

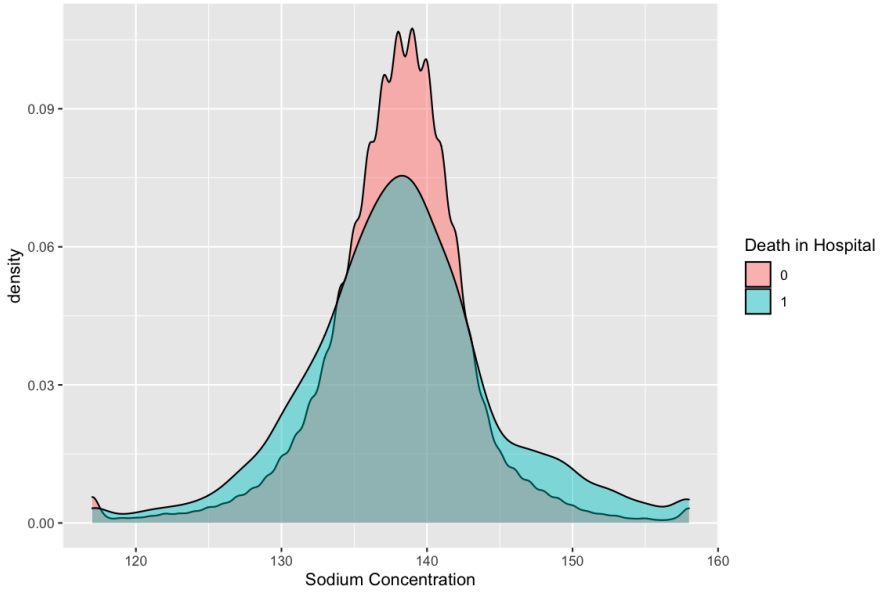

Figure 6: Sodium Concentration vs. In-hospital Mortality

Other variables, such as Sodium Concentration, a lab value, display better separation. Both at the higher end and lower end of the range there is a meaningful difference between survival and death. It is worth pointing out that the relationship between mortality and Sodium Concentration is U-shaped as opposed to linear. A linear model, such as logistic regression, would not be able to represent this pattern. Either feature engineering, to transform the variable or perhaps bin it into several categories (low, normal and high Sodium Concentration) would be necessary to more dutifully capture this relationship. This is one reason why nonlinear models, such as Random Forest or Deep Learning model, may be better at modeling this type of problem.

Pairwise Correlations

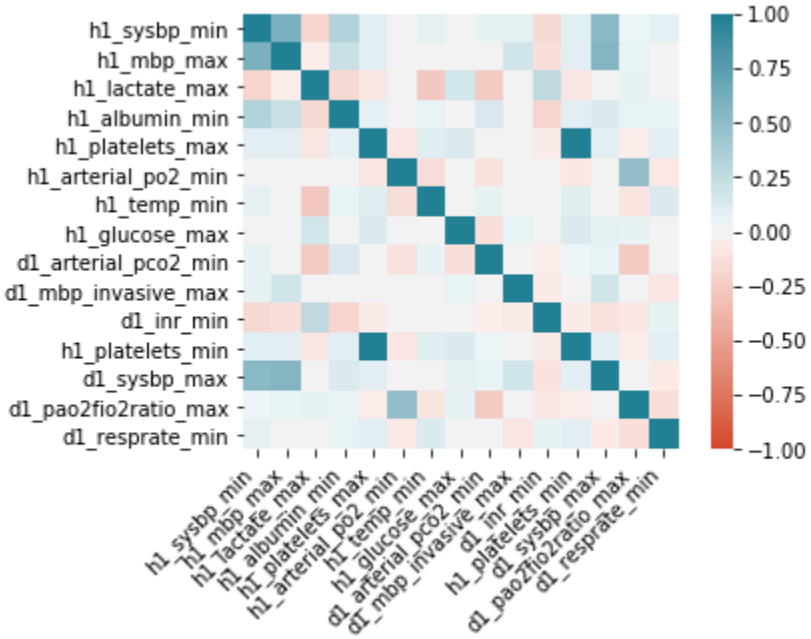

Figure 7: Correlation Plot of Random Subset of Variables

It is standard practice to perform either Principal Component Analysis (PCA) or examine correlation plots to explore linear relationships between each of the numeric variables. The complete analysis involved examining all pairwise correlations between numeric variables, though there are too many variables to comfortably display in a single plot so a random subset for fifteen numeric variables is shown here for demonstration purposes. We see a handful of high positive pairwise correlations in the range of 0.75 - 1.0 although most are not very large. There are very few large negative correlations.

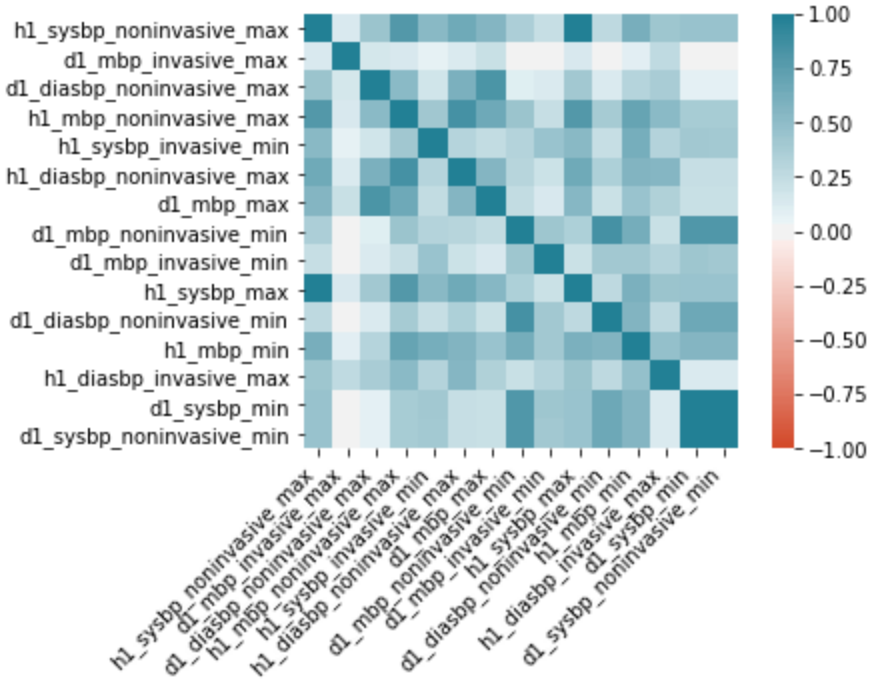

Figure 8: Correlation Plot of Blood Pressure Variables

When examining only the 36 variables related to blood pressure there is a much higher level of positive correlation. Once again, all of the variables would not fit comfortably in a single plot so a random subset of 15 blood pressure variables is shown above for demonstration. Such a high degree of correlation can be problematic for certain models that do not handle correlated variables well. In general, if the degree of correlation can be removed either through use of PCA to select only the top principal components, or simply removing a number of predictors, models tend to be more stable.

Examining Missingness

Missing values are extremely common in electronic health record datasets. There are many possible reasons for a data element to be missing, including simple transcription/entry errors, unsuccessful lab tests (if, for example the collected sample registered below the limit of detection for the test) or if a value is deliberately missing (e.g. the value for 'Heart Rate' is missing because the measurement was never taken). The first step in examining the missingness in the dataset was to plot the percentage of missingness in each variable in a histogram.

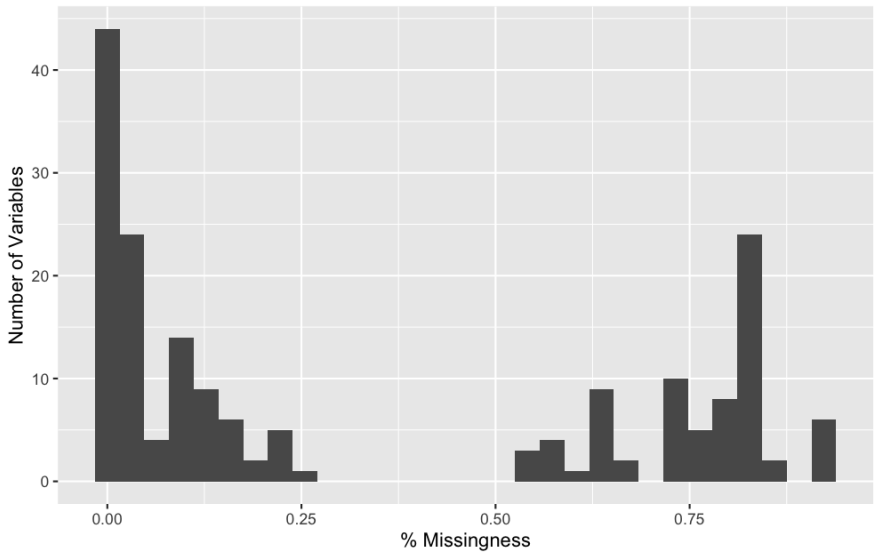

Figure 9: Histogram of Predictor Variable Missingness

Unsurprisingly, a large proportion of variables contain missing values. Only 34 variables out of 186 had less than 1% missing values. 41.2% of the variables had a missingness proportion greater than 50%. Strangely, the above distribution appears to be bimodal. That is, 41.2% of the predictor variables have missingness in the 50-92% range, and the remaining 58.8% have missingness less than 25%. Highlighting only lab value or vital sign variables compared with the remaining variables yields the following plot.

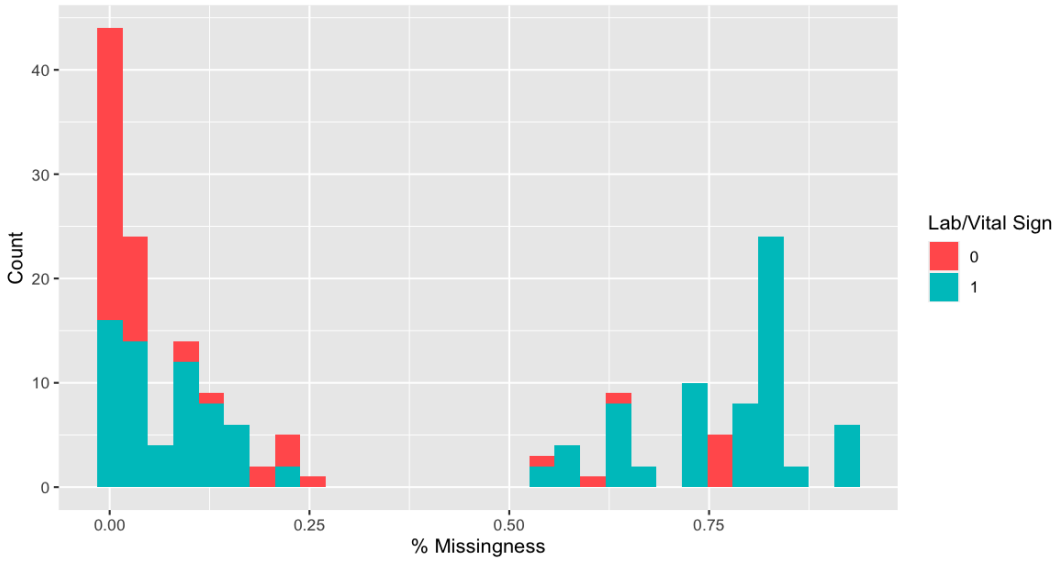

Figure10: Missing of Lab Value/Vital Signs vs Other Variables

We see that, although lab values/vital sign variables and other variables both have missingness across the entire range, lab values/vital signs tend to dominate the upper range. The top 40 variables in terms of Percent Missing are all lab values/vital sign variables. This suggests something unique about the manner in which

lab values/vital sign were collected or recorded in the dataset. This has implications for how the missing values will be handled later on in the analysis.

In order to appropriately impute, remove or ignore the missing element, it is imperative to understand the specific missingness mechanism. These fall into one of three categories. The following definitions are taken from the Handbook of Missing Data Methodology by Geert Molenberghs et al. (2015).

Missing Completely at Random (MCAR)

Data are said to be missing completely at random when the probability that responses are missing is unrelated to either the specific values that, in principle, should have been obtained or the set of observed responses.

Missing at Random (MAR)

In contrast to MCAR, data are said to be missing at random (MAR) when the probability that responses are missing depends on the set of observed responses but is further unrelated to the specific missing values that, in principle, should have been obtained.

Missing Not at Random (MNAR)

In contrast to MAR, missing data are said to be MNAR when the probability that responses are missing is related to the specific values that should have been obtained, in addition to the ones actually obtained.

Details on the specific missingness mechanism were not provided by the competition organizers so it is left to the analyst to make assumptions as best they can. In the case of the lab values/vital signs, having such a high proportion of missingness, particularly in relation to the other variables, suggests a different reason other than random chance is responsible for the missing values. It is important to understand the process of administering care for an ICU patient and how it relates to data being gathered on that patient.

Some variables, such as gender, age and other demographics, are recorded as a matter of routine for all patients that are admitted to the hospital. Others are gathered as a result of a specific test or procedure that is ordered. For example, a physician suspects a patient of having a certain condition, for which a high albumin concentration may be a symptom. The physician orders a lab test to measure that patient's albumin concentration. Depending on how rare the condition is, or how useful knowledge of a patient's albumin concentration is for diagnosing or monitoring other conditions, the lab test may be ordered more or less frequently. But the test is not ordered randomly, so we can assume that the vast majority of missing values stem from a conscious decision by the physician to not order the test.

As this example illustrates, knowledge of the process by which a lab test value is obtained is critical to ascertaining the missingness mechanism. Since gender is a standard field that is collected for every patient, the small proportion of missing values are most likely related to a transcription error or database malfunction as opposed to anything related to the specific patient encounter. However, assumptions about the remaining variables are largely ill-informed. Without a subject matter expert elucidating the pertinent details of how each variable is collected, the true missingness mechanism will remain unknown. Nevertheless, we will do our best to use reasonable assumptions.

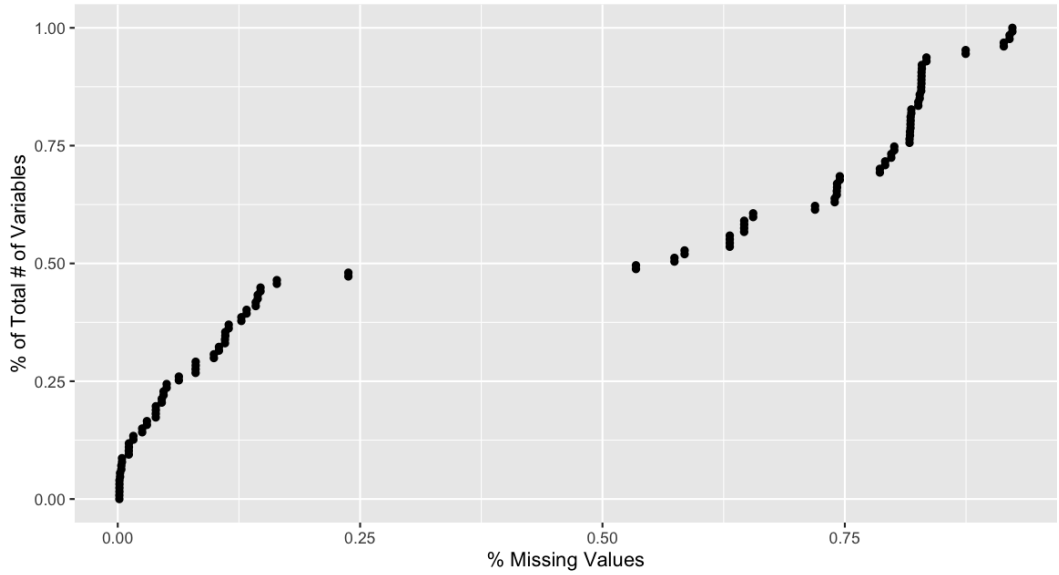

Figure 11: Percentile Plot of Lab Value/Vital Sign Missingness

Examining only the lab value variables, an interesting pattern emerges. Exactly half the variables exhibit missingness less than 25% and the remaining half exhibit missingness between 50-92%. As previously mentioned, there is almost certainly more to uncover as to why this pattern exists, but it is unfortunately beyond the scope of this project to do so.

Data Preprocessing

Impute missing values

In order to prepare the dataset for modeling a number of steps should be taken. Though some models handle missing values natively (Gradient Boosting Machines, for example) others do not. Even for models that are able to handle missing values without the benefit of preprocessing, it may be advisable to impute missing values if it can be done reasonably. Imputation will proceed based on the type of variable. As previously mentioned, we do not know the precise missingness mechanism, so some broad assumptions will be used. Lab values and vital sign variables are presumed missing not at random. All other variables are presumed missing completely at random. This has implications for the choice of imputation method.

A complete case analysis (where only records that have zero missing values are included) was considered but quickly discarded as impractical as there are only 20 complete cases in the entire dataset.

Table 4: Data Types of Lab Values/Vital Signs vs. Other Variables

	Data Type(s)	**Example**
Lab values/vital signs	*Continuous*	The highest bicarbonate concentration for the patient in their serum or plasma during the first 24 hrs of their unit stay
Other	*Integer*	Age
	Binary	Whether the patient has been diagnosed with non-Hodgkin lymphoma.
	Categorical	Ethnicity
	Continuous	The APACHE IVa probabilistic prediction of in-hospital mortality for the patient.

The predictor variables all fall under one of four categories. Lab value/vital sign variables are all continuous. The remaining variables are one of integer, binary, categorical or continuous data types.

Categorical Variables

Table 5: Missingness of Categorial Variables

Variable	# of levels	% Total of most common level	% Missingness
apache_3j_bodysystem	11	32%	1.8%
apache_3j_diagnosis	400	4%	1.2%
ethnicity	6	77%	1.5%
apache_2_diagnosis	45	12%	1.8%
apache_2_bodysystem	11	42%	1.8%

Categorical variables exhibited very low levels of missingness. In the case of ethnicity, mode imputation is a very strong case for the best imputation method. Not only does one level dominate the variable with 77% of the records, but it is unlikely that information relating to ethnicity can be constructed from other variables in the data set. That is, if we were to build a more sophisticated predictive model to impute the missing values of ethnicity, it is unlikely that model would be much better than chance.

How the remaining categorical variables should be imputed is not so obvious. The APACHE body system variable (the section of the body related to the primary diagnosis) has 11 distinct levels that are fairly evenly spread.

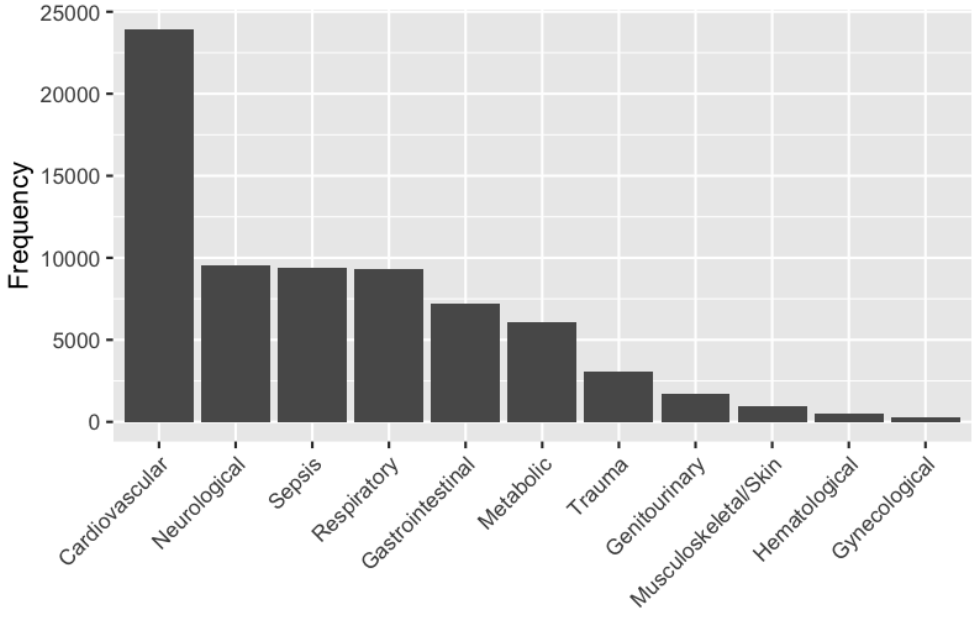

Figure 12: Histogram of Body System variable

Though 'Cardiovascular' is the most likely value for missing data, it is 68% likely to belong to a different category as well. Further, the information for which body system the diagnosis relates to is very likely contained elsewhere in the data set and a suitable predictive model for imputing missing values can likely

be constructed. However, given the low levels of missingness for the categorical variables it was decided that mode imputation would likely be sufficient. The same logic was applied to justify mode imputation for the remaining categorical variables.

Binary Variables

Table 6: Missingness of Binary Variables

Variable	**Level**		**Missing %**
	0	**1**	
aids	99.12%	0.09%	0.80%
apache_post_operative	80.04%	19.96%	0.00%
arf_apache	96.36%	2.84%	0.80%
cirrhosis	97.64%	1.56%	0.80%
diabetes_mellitus	76.96%	22.25%	0.80%
elective_surgery	81.81%	18.19%	0.00%
gcs_unable_apache	97.89%	0.96%	1.15%
male	46.10%	53.90%	0.02%
hepatic_failure	97.90%	1.30%	0.80%
immunosuppression	96.61%	2.59%	0.80%
intubated_apache	84.10%	15.10%	0.80%
leukemia	98.51%	0.70%	0.80%
solid_tumor_with_metastasis	97.17%	2.03%	0.80%
ventilated_apache	66.88%	32.32%	0.80%

Binary variables also exhibit very low levels of missingness. Additionally, many of them are highly skewed towards a single level. Mode imputation would also be suitable in this case. It was noted that many of the binary variables (e.g. ‘aids’, indicating whether the patient has been diagnosed with AIDS) can likely be predicted from other predictor variables in the dataset. Due to this consideration, it was decided to employ a predictive model approach to imputing the missing values. Specifically, KNN imputation used.

KNN imputation works by using a K-Nearest Neighbors model to impute values based on the K "nearest" (using Euclidean distance, although the algorithm can also accommodate other distance measures). From the Python Sci-Kit Learn documentation[v]:

"Each sample's missing values are imputed using the mean value from the k nearest neighbors found in the training set. Two samples are close if the features that neither is missing are close."

Lab Values/Vital Sign Variables

Lab value and vital sign variables exhibit a high degree of missingness. As examined previously, there is a strong indication the omission of a value in the data set is deliberate and the values are MNAR. As such, using a predictive model, or even a naive method like mode imputation, would be a grave mistake. Instead, a value is imputed that allows the chosen machine learning models to function, without imparting additional bias. Two different constants are imputed, creating two different training datasets. The first value is an extreme negative constant, -9999, meant to be easily distinguishable from every possible lab test or vital sign values, which are always positive. The second value is zero. The two differing training sets are subsequently referred to as the '-9999 imputed training set' and the 'zero imputed dataset'. The results of these two different methods are shown on the 'HCO3 Minimum' variable below.

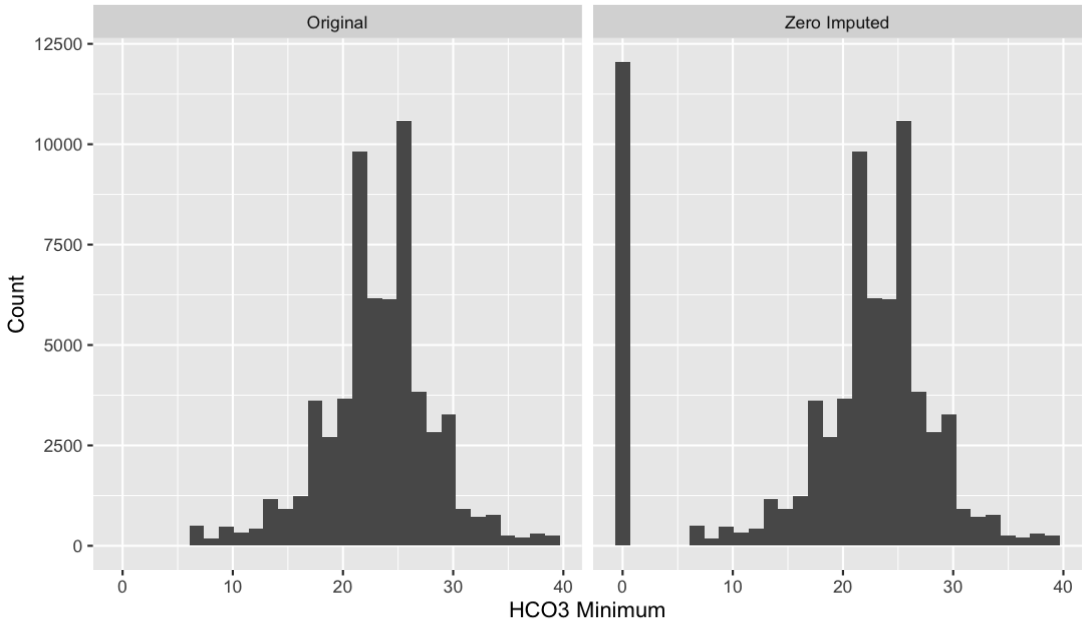

Figure 13: Effect of Zero-Imputation on HC03 Minimum

[v] https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html

Logic Checks

Missing values are one type of transcription error. Another that must also be accounted for is an erroneous value being entered into the system, either by human or machine error (e.g. a malfunction in the heart rate monitor). Though these errors are largely undetectable if the observed value is still within a reasonable range, some values are impossible or extremely unlikely and can be corrected. However, to properly perform this task subject matter expertise of the reasonable or medically feasible range of values for each individual numeric variable is needed. Instead, we are left to apply a few reasonable heuristics that hopefully are a passable approximation.

All the numeric variables were checked to ensure all the values were positive and negative values were imputed to the minimum observed value for that variable (which was zero in most cases).

The second circumstance in which we can detect an obvious error is with the paired min/max variables. Lab value variables are captured in this dataset as both the minimum and maximum values observed in the first 24 hours of admission. In the case where the minimum is larger than the maximum, the two are switched.

Feature Engineering

Creating Min/Max Variables

A small number of features were engineered to support the model in learning the important representations in the raw data in a way that would maximize predictive power. In the case of the lab value variables which contain both a minimum and a maximum value, it was decided to create a third variable calculated as the difference between the two. In some cases, a low or high lab value, by itself, is not necessarily indicative of an adverse health outcome. But, if there is a large change during a 24 hour period that may be related to mortality. There are 64 min/max pairs, resulting in the creation of 64 new features.

Binning Diagnosis Variable

The second feature engineering step was to bin the primary diagnosis variable to reduce it from 397 unique values to 112. Many of the levels had only a very small number of cases, so it is unlikely that the model can learn what an appropriate level of mortality risk is represented by the presence of that diagnosis. If the values can be binned together in a way that preserves as much information as possible it will likely boost the predictive power of the model.

Definitions for each possible value of the APACHE diagnosis code were obtained from ANZICS "Adult Patient Database Data Dictionary" (Version 5.8, Nov 2019). Groupings are already provided and at face value appear to work quite well for the task of grouping values by severity of the diagnosis.

703	703.01	Alcohol withdrawal
	703.02	Drug withdrawal
	703.04	Overdose, alcohols (ethanol, methanol, ethylene glycol)
	703.05	Overdose, analgesic (aspirin, acetaminophen)
	703.06	Overdose, antidepressants (tricyclic, lithium)
	703.07	Overdose, other toxin, poison or drug
	703.08	Overdose, sedatives, hypnotics, antipsychotics, benzodiazepines
	703.09	Overdose, street drugs (opiates, cocaine, amphetamine)
	703.10	Envenomation by snake
	703.11	Envenomation by jellyfish and other invertebrates
	703.12	Envenomation by other animal

Figure 14: Diagnosis codes for withdrawal, overdose and envenomation

For example, diagnosis codes 703.01-703.12 can be grouped together by rounding to 703. Each diagnosis code is relatively similar to the others in terms of pathology, recommended course of treatment and mortality.

1302	1302.01	Thoracotomy for benign tumour (e.g. mediastinal chest wall mass, thymectomy)
	1302.02	Thoracotomy for lung cancer
	1302.03	Thoracotomy for other malignancy in chest

Figure 15: Thoracotomy diagnosis codes

Thoracotomy variables may not be as easily grouped. For instance, a thoracotomy for a malignancy or lung cancer may be more serious than a benign tumor. Medical expertise is needed to properly evaluate the extent to which this is true, which, as mentioned is out of scope for the project. A manual review of the 112 unique groupings appears to be a reasonable starting point and is likely an improvement over the original 397 unique levels.

Binning Other Variables

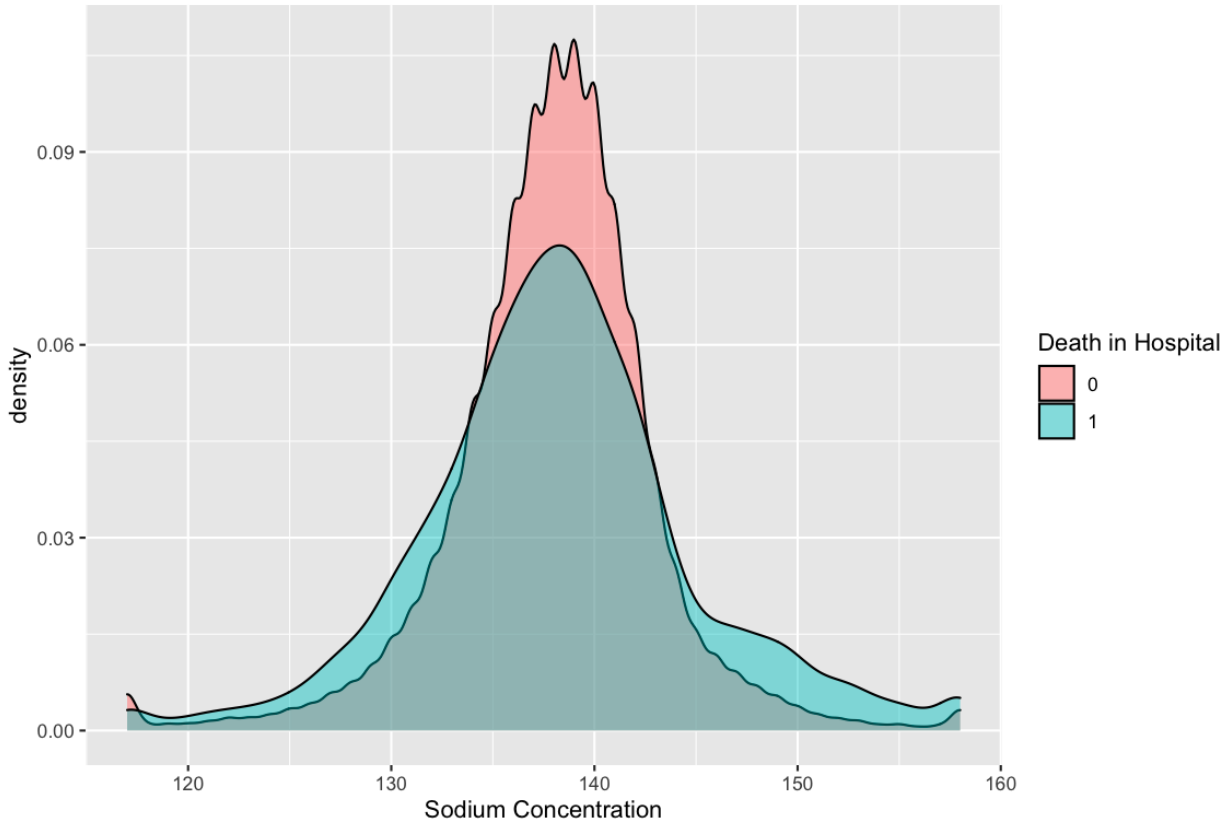

Figure 16: Sodium Concentration by Mortality

As seen earlier, it seems likely that many variables observe a U-Shaped relationship to the outcome variable, as opposed to a linear relationship. In order to facilitate the models learning this relationship, it may be helpful to bin the continuous variable into discrete levels, such as low, medium and high. This way the model can learn that a medium range measurement of sodium concentration has a low probability of death compared with high or low values. However, Kuhn & Johnson (2018) advise against binning for most machine learning models as information is lost whenever this is done. Most machine learning models are able to learn the non-linear relationship between the predictor and the outcome. Decision tree based models, such as Random Forests or XG Boost, are perfectly capable of learning non-linear relationships like the one above. A logistic regression model, however, is not able to learn this type of relationship and this may be one of the reasons why it does not perform as well as some of the other models used in this analysis.

Given that most models are able to handle these continuous variables natively without additional feature engineering, no binning was performed allowing for the maximum retention of information in each.

Remove Highly Correlated Variables

Many of the numeric variables in the dataset are highly correlated as they are slightly different variants of the same type of measurement. For example, 'Mean Diastolic Blood Pressure, invasively measured' is highly correlated with 'Mean Diastolic Blood Pressure, non-invasively measured.' Additionally, there are 64 new min/max variables which are likely to be highly correlated with the min/max features that were used to create them. This can be problematic for a number of reasons. First and foremost is that many models become unstable in the presence of multicollinearity. The second is that parsimonious models are generally preferred where performance is similar to models with a greater number of parameters. In the machine learning context, the probability of being trapped in a local minimum, as opposed to the global minimum, increases with the number of parameters used to train the model.

The 'findCorrelation' function in R's Caret package allows the user to set a threshold for the maximum pairwise correlation and then returns a vector of the variables to be removed in order to have all remaining pairwise correlations fall below this level.

Table 7: Percentage of removed variables at different correlation thresholds

Threshold	**% of Variables to Remove**
0.9	28.3%
0.95	25.1%
0.975	18.0%

In order to achieve a maximum pairwise correlation (meaning all remaining pairwise correlations are equal to or less than 0.975) 18% of the predictor variables would need to be removed. This percentage increases when the threshold is lowered. There is a tradeoff between removing predictor variables to improve model stability and retaining as much information in the dataset as possible. It was decided to set the threshold to 0.95 and remove the 25.1% of variables recommended by the algorithm.

Ideally, which variables were removed would be informed by a subject matter expert. There is likely a medically-motivated reason for preferring one variable over another variable that is 98% correlated with it, but that is beyond the scope of this analysis. Another approach would be to retrain each model on all three thresholds and select the threshold with the best performance. However, this is precluded by the limits on computation power and time.

Removing Highly Skewed Variables

Along similar lines for preferring parsimonious models, another Caret function was applied to identify variables that are highly skewed towards a single value and remove those variables from the

training set. 'nearZeroVars' is a function that "diagnoses predictors that have one unique value (i.e. are zero variance predictors) or predictors that are have both of the following characteristics: they have very few unique values relative to the number of samples and the ratio of the frequency of the most common value to the frequency of the second most common value is large." 36 such variables were identified. The default ratio of 95/5 for the most common to 2nd most common unique values was used. 10 of the 36 variables were binary predictors. The AIDS variable, for example, had 99.1% a single value.

Table 8: Skewness of AIDS variable

	Overall (N=73370)
AIDS	
0	72721 (99.1%)
1	64 (0.1%)
Missing	585 (0.8%)

The remaining 26 variables were all min/max variables. That is, they were created by subtracting the maximum variable from its corresponding min variable. The reason this difference resulted in a highly skewed feature for many of the min/max pairs is that the min and max values were often the same, resulting in zero as the most common value. This is exemplified by the Lactate Min/Max variable shown below.

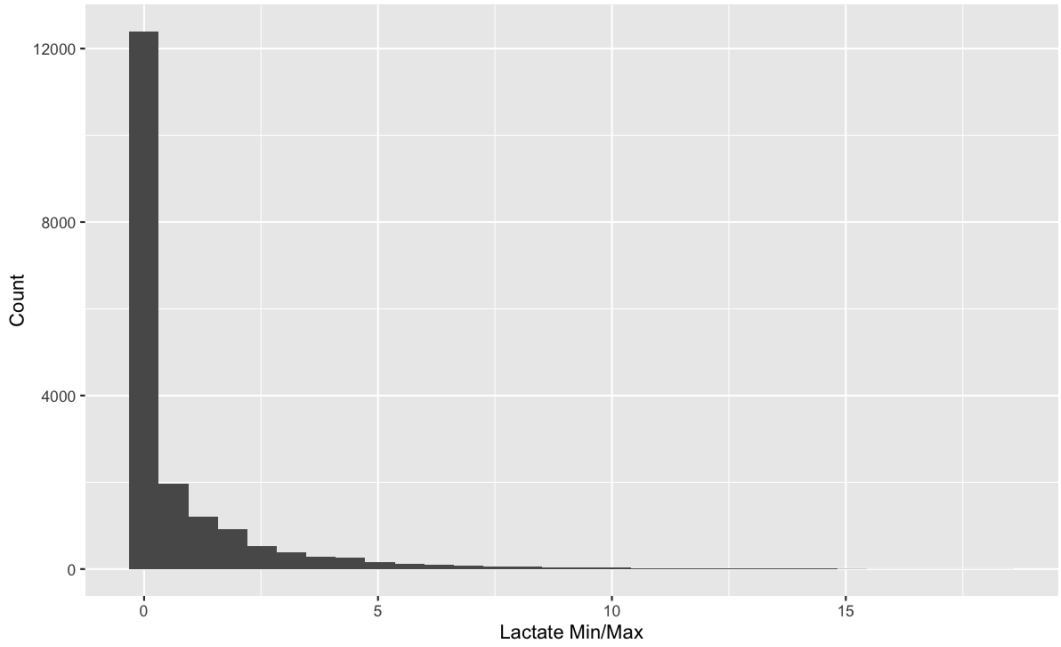

Figure 17: Histogram demonstrating skewness of Lactate Min/Max

One Hot Encode Categorical Variables

Most machine learning and statistical models do not handle text inputs. As such, any categorical variables must be transformed into a numeric format before modeling. In the case of nominal features, where there is no inherent order to the elements, one-hot encoding is preferred.

Center and Scale Numeric Variables

A final step to pre-process the data before it is used to train a machine learning model is to standardize all the numeric variables. While many models are invariant under linear transformation of the predictor variables, others, such as deep learning models, are not. Linear regression, for example, is unaffected if a predictor variable is represented in litres or milliliters. It will change the interpretation of the coefficients but will not change the output of the model. Others, however, are not invariant under such conditions. Centering and scaling ensure that a particular variable is not weighted more or less depending on the arbitrary choice of scale.

To center and scale, the mean and standard deviation of each numeric variable was calculated on the training set only. Then each value had the mean subtracted and was divided by the standard deviation. The second step was then performed on both training and test sets.

Modeling

Objectives

Several objectives guided the steps taken during data pre-processing and modeling phases.

1) *The primary outcome metric for training and evaluating each model is Area Under the Receiver Operating Characteristic Curve (AUC).*

There are at least a dozen different popular metrics to evaluating the performance of a binary classifier. AUC is the most common choice in the reviewed literature and in general is a widely used metric. AUC strikes a balance between maximizing sensitivity (the proportion of positive cases the model is able to detect) and specificity (the proportion of negative cases that are properly identified). This is in contrast to a metric like accuracy, which simply seeks to maximize the number of properly identified cases, either positive or negative. If accuracy were used, a seemingly high performing model can be constructed simply by always predicting survival. Since the outcome variable is skewed to 90% negative cases, this naive model would be correct 90% of the time and achieve a relatively high accuracy. AUC fixes this issue by placing an equal priority on identifying negative cases correctly as well. In training the models through use of k-fold cross validation, AUC was used as the evaluation metric.

2) *Avoid overfitting the training dataset.*

Given the extremely large solution space available to machine learning algorithms, it is a common occurrence to overfit a model to the training set. Though it is not always possible to prevent, it was a foremost consideration that the final model generalizes well to the test dataset. To this end, k-fold cross validation was used, and certain feature engineering steps were avoided. Instead, only features for which there was a plausible justification for the causal relationship between the feature and the outcome variable were included.

3) *Properly utilize the available computation power*

Compromises had to be made in terms of which models were trained and how many parameter sets were able to be fit. A more effective approach would be to train the models on a machine learning server but this was unfortunately not feasible. Instead, a narrow but reasonable parameter set for each model type

was tested, and where time permitted, further parameter sets in the neighborhood of the most promising parameter set were also tested.

Resampling Strategy

All models were trained using 10-fold cross validation. This works by splitting the training set in 10 equally sized subsets. A model is trained on 9 folds and evaluated on the 10^{th}. This is repeated ten times so each subset is used for evaluation once. In this way, we can gain some understanding of how the model would perform on unseen data.

The same folds were used on each of the four different model types that were trained. This is a necessary step to be able to compare different model architectures. If an arrangement of the data were used to train a Random Forest model that was used for a Support Vector Machine, the difference in performance might be attributed to differences in the model architecture, or differences in the sample used. In order to isolate the former, the same partition of the data was used to train and evaluate each model.

Multiple repetitions are often used in k-fold cross validation to account for the bias that may be due to the particular arrangement of the folds. Each repetition creates a different partition of the folds and hence limits this bias. Due to the high computation burden of this approach however, only a single repetition was used.

Overview of Model Architectures

Four different models were trained and evaluated: Penalized Logistic Regression, Random Forest, XG Boost and Support Vector Machines (SVM). Somewhat coincidentally, these are the top four most popular models as seen in the surveyed literature. Using a Neural Network was considered but ultimately rejected for two reasons: they are computationally expensive to train and are unlikely to perform as well on low dimension, tabular data such as this.

Penalized Logistic Regression

Logistic regression is the default method for binary classifiers and has enjoyed widespread use for decades, predating the rise of machine learning methods. It is a type of generalized linear model that has foundations in classical statistical theory. As datasets became larger and larger, the risk of overfitting by including variables with only a small or even non-existent relationship to the outcome variable increased. To minimize this issue, two different methods have become popular. Lasso regression, which imposes an L1 penalty on the regression coefficients, and Ridge Regression, which imposes an L2 penalty. The elastic net was proposed by Hastie & Tibshirani in 2003 and seeks to combine the best part of the two methods. It

adds a mixing constant, which specifies the proportion of each penalty to use. The effect is a more robust method than often outperforms either ridge or lasso individually.

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|$$

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2$$

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\left[\frac{1}{2}(1-\alpha)\sum_{j=1}^{p}\beta_j^2+\alpha\sum_{j=1}^{p}|\beta_j|\right]$$

Figure 18: Residual sum of squares formula for lasso, ridge and elastic regression.

The two parameters to tune are:

α = mixing proportion

λ = shrinking parameter.

The mixing proportion specifies how much of each penalty to use. $\alpha = 1$ simplifies to lasso regression, $\alpha = 0$ to ridge regression. λ has the same interpretation in elastic net as in ridge or lasso regression. The two parameters are usually tuned using cross validation.

Random Forest

Random Forest is a tree-based model that utilizes bagging, also referred to as bootstrap aggregation. Bagging is the process of taking many bootstrap samples, training a model on each sample, and then aggregating the models in some way. In the random forest context, each bootstrap sample is used to build a single decision tree using the data. This process is repeated for many bootstrapped samples and the resulting set of decision trees are aggregated. If the objective is regression, then the output of each decision tree is added up and average to produce a single number. In classification, each tree 'votes' for a particular outcome and the number of votes is added up and divided by the total to produce a probability for that outcome.

Bagging is a technique that works best with high variance, low bias methods such as decision trees. Trees are in many ways ideal candidates for bagging, since they can capture complex relationships and interaction structures in the data. If they are allowed to grow to a sufficiently deep size, then they also exhibit very low bias. The averaging of many individual trees is effective at reducing variation while maintaining the low bias.

Generating bootstrap samples introduces a random component into the tree building process. This produces a *committee* of decorrelated trees, though, in practice, the trees are not completely independent since the same set of original predictors is used to build each tree. This is in contrast to boosted trees, where subsequent trees are built to focus on the particular records that were poorly predicted up to that point. This negatively correlates new trees with previous ones and guarantees that the trees are not independently distributed.

Since bagging alone is not enough to achieve a sufficiently high level of decorrelation, Random Forests add in additional randomness to the algorithm. For each split of the decision tree, only a subset of the original predictors are available. The tree then selects the best predictor from the subset to split on.

```
Select the number of models to build, m
for i = 1 to m do
    Generate a bootstrap sample of the original data
    Train a tree model on this sample
    for each split do
        Randomly select k (< P) of the original predictors
        Select the best predictor among the k predictors and
        partition the data
    end
    Use typical tree model stopping criteria to determine when a
    tree is complete (but do not prune)
end
```

Figure 19: Basic Random Forest Algorithm. Kuhn & Johnson (2018).

One of the useful features of random forest (as well as Gradient Boosting Machines) is they often have built in feature importance functions. Feature importance is calculated by how much a predictor variable reduces the mean Gini impurity across the ensemble of trees.

Random Forest also has a built-in cross validation structure. Many implementations, including those in Python and R, are able to calculate 'out of bag' (OOB) error; the error calculated on samples not included in the bootstrap sample. According to Hastie et al. (2013), "OOB is the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample."

In the Scikit-learn implementation of Random Forest there are nineteen parameters that can be set. The most important and the focus of the tuning process are:

Max features = Number of randomly selected predictors at each split (max features?)
Number of Estimators = Number of trees to construct
Max depth = The maximum depth of each tree

Restricting the max features is one way to increase the independence of the ensemble. By using a small number of predictors relative to the total number available, each tree has a better chance of being different from preceding trees. Maximum depth controls how complex an individual decision tree can be. Increasing the depth reduces bias but increases variance.

Because trees in a random forest are independent, they are resistant to overfitting when a large number of trees are used. However, they can be computationally burdensome, so it is often more practical to start with a relatively small number and stop adding trees when cross validation error plateaus.

XGBoost

XGBoost is a tree-based model built under the gradient boosting machine framework.
It has risen to prominence in the past four years as it is famously the preferred algorithm used by many of the top Kaggle competition winners. XGBoost stands for "Extreme Gradient Boosting", where the origination of gradient boosting is taken from the landmark paper 'Greedy Function Approximation: A gradient boosting machine" by Friedman (1999). According the Brownleee J. (2018), Gradient Boosting Machines (GBMs) contain three components.

1) A loss function to be optimized
2) A weak learner to make predictions.
3) An additive model to add weak learners to minimize the loss function. (p. 10)

The loss function depends on the type of problem being solved. Squared error or mean absolute error are common choices for regression and logarithmic loss is common for classification problems. The XGBoost algorithm also accommodates custom loss functions defined by the user.

The weak learners used in GBM's are decision trees. This takes the idea of Ada Boost, which uses stumps, or decision trees with a single node, one step further and allows for deeper and more complex trees to be built.

Trees are added one at a time and subsequent trees are built in such a way to focus on parts of the dataset which are poorly predicted by previous trees. In this way, past trees influence the construction of future trees and are hence not independent of each other. Training stops when a fixed number of trees are built, or the loss reaches an acceptable level or no longer improves on an external validation set.

Given that GBM's are greedy algorithms which can quickly overfit, a number of regularization methods are used to prevent this from happening. These include constraints on tree size/complexity, the learning rate or 'shrinkage', the subsample of the training data used at each step or a regularization weight (such as L1 or L2) imposed upon the trees.

Support Vector Machines

Support Vector Machines (SVMs) were first developed in the mid-1960's by Vladimir Vapnik and have since evolved into a flexible and widely used machine learning algorithm for both regression and classification. SVMs work by constructing a hyperplane in high dimensional space that achieves the maximum separability of the two classes (called the 'margin'). The margin is the distance between the classification boundary and the closest data point in each class.

> "An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall SVMs can perform both linear and nonlinear classification. By using different kernels, the SVM essentially maps its inputs into a different high dimensional feature space." [vi]

[vi] https://en.wikipedia.org/wiki/Support_vector_machine

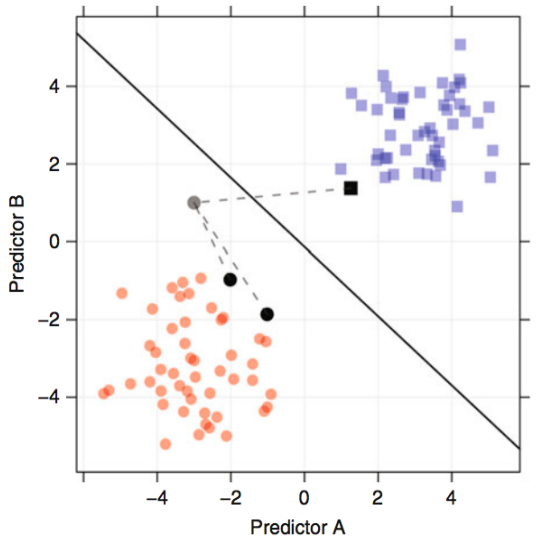

Figure 20: SVM Boundary for Binary Classification Problem Kuhn & Johnson (2018).

Shown above is a binary classification problem with two predictor variables. The black line indicates the separation boundary that maximizes the distance from the nearest data points. New points (the grey dot) are predicted by plotting them in space and calculating which side of the decision boundary they fall under.

Model Fitting Procedure

Each of the four models were fit using the same procedure. A set of plausible values was determined for the most important parameters and exhaustive grid search and 10-fold cross validation was used to train models for each permutation of parameter values. Exhaustive grid search is the process of fitting a model for every possible permutation of parameter values. This is in contrast to random grid search, where a fixed number of parameters sets are chosen at random from a larger set of possible values. Due to the high computation burden of this approach, a limited subset of parameters and possible values was used. For many parameters, only a single value was used.

In the case of the SVM model, the time to train a model was so large it precluded testing more than two parameter values. XGBoost, while faster to train, has a large number of parameters so only the four parameters that were deemed the most important could be incorporated into the parameter grid.

Table 9: Random Forest Parameter Grid #1

Parameter	**Values**
Number of Estimators	(8, 10, 12, 50)
Max Tree Depth	(2, 4, 8)
Max Features per split	(20, 50, 100)

This results in 3x4x3 = 36 permutations of parameter values. Each permutation is trained using 10-fold cross validation resulting in 360 models being trained.

In some cases, after training with the initial parameter grid is complete, a second grid is created that focuses on the most promising section of the parameter space. In the Random Forest example, the best performing parameter set was:

Number of Estimators = 50

Max Tree Depth = 8

Max Features per split = 50

A second parameter grid was made to explore values around this set, resulting in an additional 3x4x3 = 36 models to be trained.

Table 10: Random Forest Parameter Grid #2

Parameter	**Values**
Number of Estimators	(100, 150)
Max Tree Depth	(8, 10, 12, 14)
Max Features per split	(50, 100, 150)

Selecting the Final Training Set

Prior to fitting the four models, an experiment was performed to select which of the two versions of the dataset with which to proceed. These two datasets correspond to the method used to impute missing lab values/vital signs. One method imputed zero, the other imputed -9999. As it takes a prohibitively long period of time to train the four models using the procedure described above on both datasets, it was decided to fit a Random Forest to both datasets and then select the one with higher performance.

The experiment used the following parameter grid:

Table 11: Training Set Selection Parameter Grid

Parameter	**Values**
Number of Trees	(50, 100, 150)
Max Tree Depth	(2, 4, 8)
Max Features per split	(20, 50, 100)

The -9999 imputed set had a peak AUC of 0.867 whereas the zero imputed dataset had peak AUC of 0.881. The zero imputed set was therefore selected and used for the remaining model fitting.

Selecting Final Models

After completing cross validation training across all the parameters sets of interest, the final set of parameter values for each model type was selected and a final model for each of the four model types was trained on the entire training set. Parameter values were selected primarily based on highest mean AUC during 10-fold cross validation, though model complexity was also considered.

Parsimonious models are in general recommended in the world of statistics as well as predictive modeling. Many of the information criterion used in statistics for the purposes of model selection are designed to penalize the addition of parameters (AIC, BIC for instance). This is to avoid overfitting to the training data. Kuhn et al. (2018) recommends training a variety of models and then choosing the simplest model that is within one standard error (of whichever cross-validated performance metric is being used) of the best performing model.

Where two models have similar performance, but one is less complex, the simpler model should be selected. In our case, since we are selecting parameter values within the same model type, this manifests as selecting some parameter values that result in a simpler model. For example, a Random Forest with fewer trees, or with smaller tree depth, is considered a simpler, more parsimonious model compared to the alternative. However, determining which parameter set constitutes a "simpler" model is not always straightforward. In the case of linear regression, it is reasonable to claim that a model with fewer parameters is simpler than a model with more parameters. But in the Random forest example, where multiple parameter types are involved, the determination is more subjective. Therefore, we are unable to use perfectly objective criteria to choose the simplest model and will use subjective criteria instead.

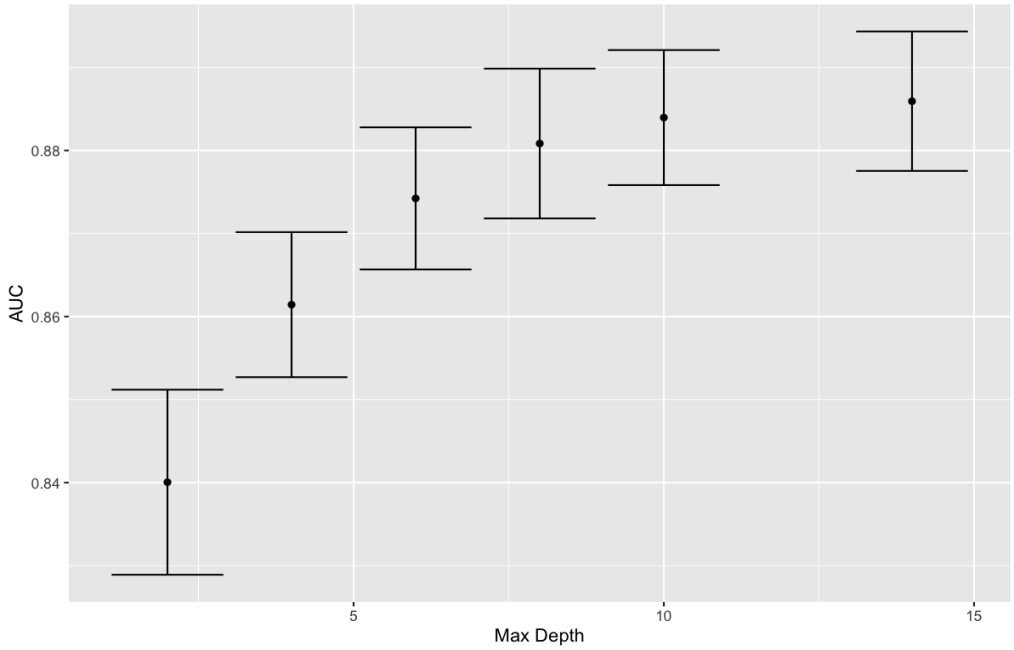

Figure 21: Standard Errors of AUC by Max Depth

The above plot shows the AUC and cross validation standard errors of six different random forest models. Each had the same fixed value for the 'max features' and 'number of estimators' parameters. The 'max depth' parameter was set from 2 to 14. The highest mean AUC was 0.886 corresponded to a max depth of 14. However, a max depth of 8 had mean AUC of 0.88, which is within one standard error of 0.886. On the grounds of parsimony, we would select the model with a max depth of 8.

Descriptive plots comparing the different parameters sets were used to better understand the impact of changing a particular parameter. This is particularly important when considering the problem is in four dimensions (3 parameters + 1 outcome variable).

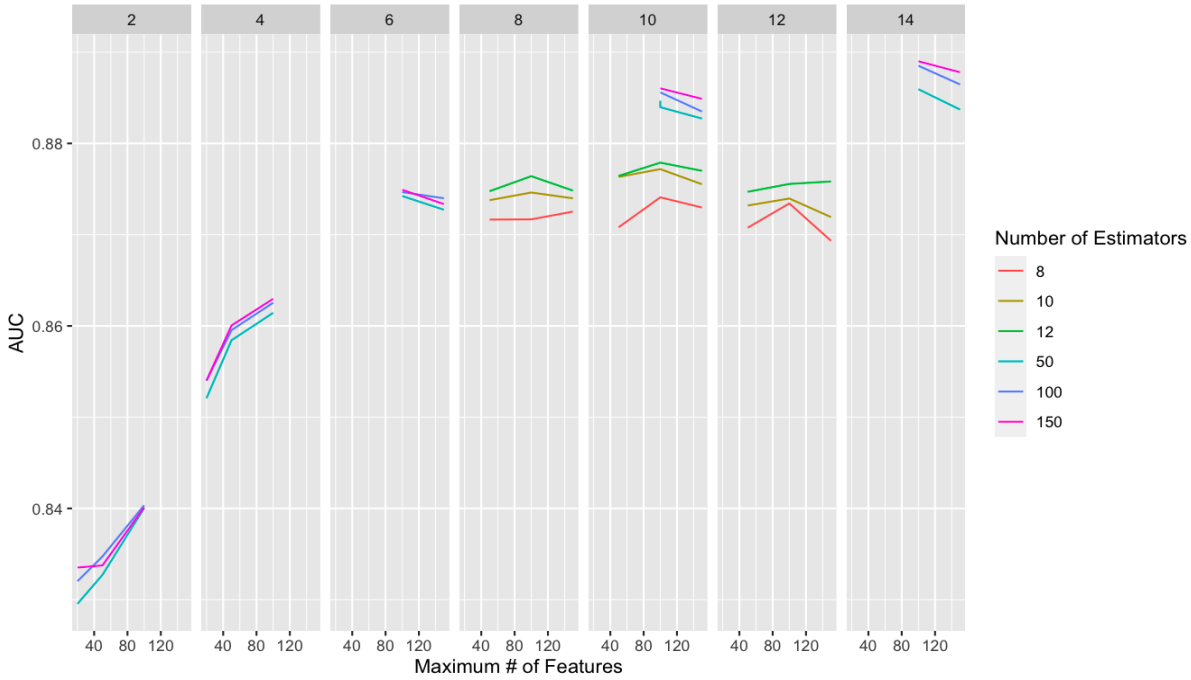

Figure 22: AUC of Random Forest Models by Max Depth, Max Features and # of Estimators

The above graph plots the outcome variable, AUC, against the three parameters of interest in training a Random Forest model. A number of interesting trends emerge when examining the plot:

- Models with a small number of estimators (8, 10 or 12) perform as well or better than models with more estimators, provided the max depth is quite shallow (2, 4 or 6). But as the depth is increased, models with a greater number of estimators start to dominate. This trend may persist even beyond a max depth of 14.
- Among models with 50, 100 or 150 estimators, models with 150 estimates tend to perform the best. This is especially true when max depth is 10 or 14.
- Among models at the higher end of the performance spectrum (AUC > 0.88) more estimators is better.

An important caveat is that, while not shown in the above plot, standard errors are usually quite small at around 0.01, which is enough to cast doubt over several of the above claims.

Given the above points, there may be opportunity to further refine the Random Forest model by expanding beyond a max depth of 14 or number of estimators equal to 150. However, significant computing resources had already been devoted to training the many models shown here, so instead we will select the best model from the candidates already assembled.

Random Forest Model Selection

Table 12: Highest AUC parameter set for Random Forest model

Parameter	**Value**
Number of Trees	150
Max Tree Depth	14
Max Features per split	100
Mean AUC (std. error)	**0.889 (0.0081)**

The best performing model was also one with very high complexity. Filtering the remaining models to those with a mean AUC greater than (0.889 - 0.0081) = 0.8809 yields the following plot.

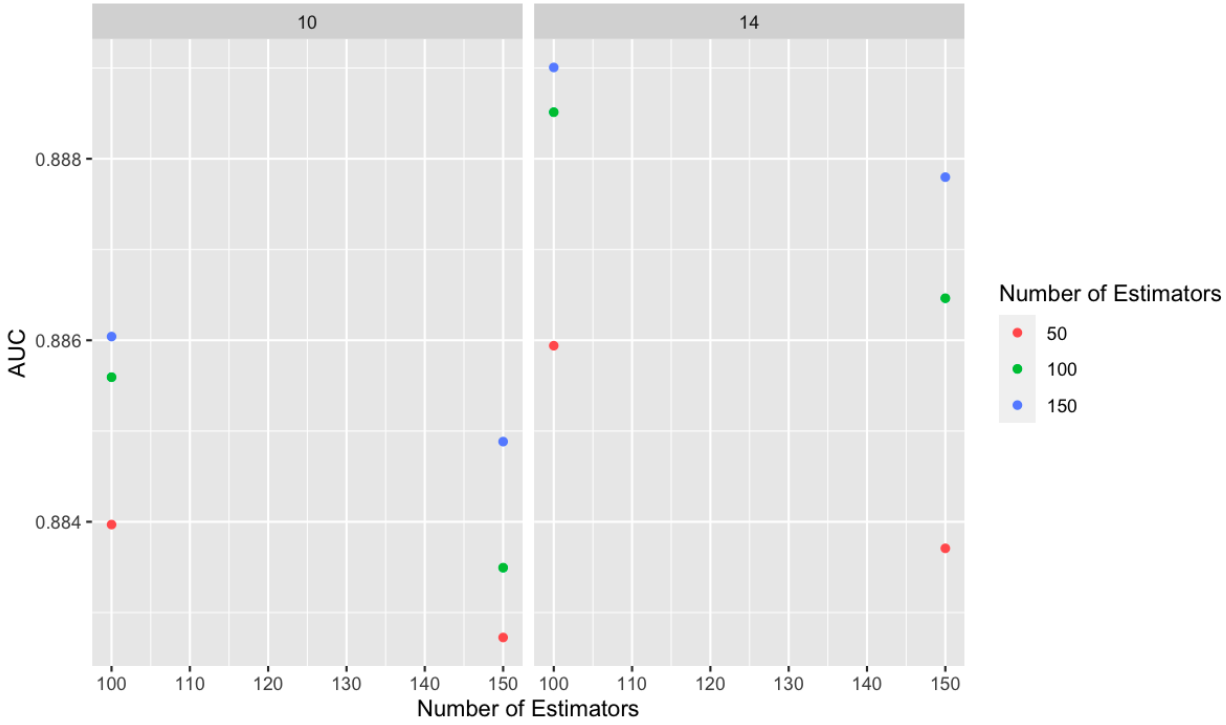

Figure 23: Candidate Random Forest models after filtering

The simplest model from this subset, with the minimum value for all three parameters, is:

Table 13: Final parameter set for Random Forest model

Parameter	**Value**
Number of Estimators	50
Max Tree Depth	10
Max Features per split	50
Mean AUC	**0.884**

This parameter set was then used to train a final random forest model on the entire training set.

SVM Model Selection

Model selection for the SVM was much simpler due to a greatly reduced number of candidate models. The time to train a single model (with 10-fold cross validation) was in excess of 48 hours. As a consequence of this, only two models were trained. Both were trained using a linear kernel. Though it would have been preferable to test a wider variety of parameter values as well as a different kernel. The two results, corresponding to the two different values used for the cost parameter, are below.

Table 14: Results of parameter tuning of SVM model

Cost	**Mean AUC (std. error)**
0.1	0.842 (0.021)
0.01	0.855 (0.010)

Based on this, the cost parameter for the final model was chosen to be 0.01 and a linear kernel was used.

XGBoost Model Selection

Table 15: XGBoost parameter grid

Parameter	**Values**
Learning Rate	(0.01, 0.05)
Max Tree Depth	(4, 8, 12)
Subsample ratio of rows	(0.3, 0.5)
Subsample ratio of columns	(0.4, 0.6, 0.8)

In contrast to the parameters set being selected for the Random Forest model, it is not clear which permutations of the XGBoost model parameters indicate a simpler model, with the exception of tree depth. Therefore, no attempt was made to apply the ‘simplest model within one standard error of the best model’ method. Instead, the parameter set with the highest mean AUC was selected for the final model.

Results

Prediction on Test Dataset

All four final models were then used to predict the hold out test dataset.

Table 16: Results of final models used to predict on test dataset

Model	Accuracy	Sensitivity	Specificity	AUC
Random Forest	0.931	0.265	0.994	0.893
XGBoost	0.934	0.329	0.991	0.900
SVM	0.926	0.203	0.995	0.861
Penalized Logistic Regression	0.919	0.068	0.999	0.828

The XGBoost model achieved the highest AUC value on the test dataset, followed closely by the Random Forest model. The SVM was 3rd and the penalized logistic regression model was last. More attention is given to the implications of this in the following Discussion section.

Feature Importance

Examining which features are most important for each model shed some light onto the underlying processes that govern the relationship between the outcome and predictor variables. Alternatively, if a model is being trained to be deployed in a production environment, feature importance can be used to eliminate predictor variables and lighten the computation burden of deploying the model.

Random Forests and XG Boost have feature importance capabilities built into the Scikit-learn implementation in Python. Logistic regression is naturally more interpretable, as the size of the individual coefficients is synonymous with variable importance as the model is not obscured by a complex structure. SVM does not have a built-in importance function, so will not be considered in this portion of the analysis.

Feature importance in XGBoost and Random Forest models are calculated in a similar manner, given that they are both tree-based models. Importance is calculated for a single decision tree by the amount that each attribute split point improves Gini Impurity (by default, although a different performance metric could also be used), weighted by the number of observations that node is responsible for.

Table 17: Top ten most important features for each model

Random Forest	XGBoost	Logistic Regression
apache_4a_icu_death_prob	apache_4a_icu_death_prob	apache_4a_icu_death_prob
d1_lactate_max	elective_surgery	d1_lactate_min
d1_lactate_min	ventilated_apache	d1_lactate_max
gcs_motor_apache	apache_2_diagnosis_114.0	ventilated_apache
d1_spo2_min	apache_3j_bodysystem_Metabolic	
d1_sysbp_noninvasive_min	apache_3j_diagnosis_102.0	
apache_2_diagnosis_114.0	apache_2_diagnosis_308.0	

gcs_eyes_apache	d1_lactate_min
d1_bun_min	apache_3j_diagnosis_703.0
d1_heartrate_min	gcs_motor_apache

The Random Forest and XGBoost models shared four of the top ten features. Random Forest had a greater preference for lab value or vital sign variables with six, whereas XGBoost preferred several of the diagnosis variables (four in comparison to just one for the Random Forest model).

Curiously, the logistic regression model used only four variables in its final model! Examining the individual coefficients yields another interesting finding.

Table 18: Coefficient values for Penalized Logistic Regression model

Coefficient Value	**Predictor Variable**
0.4045	apache_4a_icu_death_prob
0.1003	d1_lactate_min
0.0778	d1_lactate_max
0.0017	ventilated_apache

The APACHE probability of death variable was weighted four times higher than the next highest predictor variable. It would appear the penalty applied to the model removed nearly all the predictors and of the ones that remainder, a single predictor carries most of the predictive power. Given this, is it likely that the penalized regression model is only a modest improvement on the given mortality prediction generated by the APACHE algorithm.

Discussion

XGBoost

The XGBoost performed the best, though its biggest advantage is not apparent simply by observing AUC. XGBoost had highest sensitivity, with only a modest decline in specificity in comparison to the second best model. Given that many of the real-world applications involve identifying patients who are at increased risk of death, this is an important consideration.

Random Forest

Random Forest had the second-best AUC, only slightly less than the XGBoost model. However, given that much more training time was allotted to tune the Random Forest parameters in comparison to XGBoost, it is possible that this gap could be further increased were equal attention to be paid in future efforts.

Support Vector Machine

While performance was similar, albeit less, in comparison the two tree-based models, the SVM took significantly longer to train. 23.9 hours elapsed to train final model compared with under 30 mins for other three models. Generating new predictions on the test set took longer as well – 104.2 sec compared with 1.84 seconds for XGBoost or 0.55 seconds with Random Forest. This would prove to be an important drawback were the goal to deploy the model in production and computing resources were limited.

Penalized Logistic Regression

The Penalized Logistic Regression model shrunk nearly all the predictor variable coefficients to zero, with the exception of only four variables. In fact, the APACHE mortality variable was especially critical to this model, with a coefficient four times larger than the next largest. If the model we retrained without this variable present, it is likely to experience a significant decrease in performance. The model did not perform well because many of the relationships between the predictor variable and the outcome are nonlinear. For example, either a high or low oxygen saturation level is indicative of increased mortality risk. However, a linear relationship would demand that if a high value was indicative of higher risk and a nominal value was minimal risk then a low value would be even less risk. This is simply not the case with many, *in fact nearly all*, of the predictor variables contained in the dataset.

The model also has extremely poor sensitivity, predicting nearly all cases to be negative. Despite high accuracy and relatively high AUC, it performs only slightly better than a naïve model which predicts survival for every case.

References

1. Awad, A., Bader-El-Den, M., McNicholas, J., & Briggs, J. (2017). Early hospital mortality prediction of intensive care unit patients using an ensemble learning approach. *International Journal of Medical Informatics (Shannon, Ireland)*, *108*, 185–195. https://doi.org/10.1016/j.ijmedinf.2017.10.002
2. Meyer A, Zverinski D, Pfahringer B, et al. Machine learning for real-time prediction of complications in critical care: a retrospective study. *Lancet Respir Med*. 2018;6(12):905-914. doi:10.1016/S2213-2600(18)30300-X
3. Meiring, C., Dixit, A., Harris, S., MacCallum, N., Brealey, D., Watkinson, P., Jones, A., Ashworth, S., Beale, R., Brett, S., Singer, M., & Ercole, A. (2018). Optimal intensive care outcome prediction over time using machine learning. *PloS One*, *13*(11), e0206862–. https://doi.org/10.1371/journal.pone.0206862
4. Delahanty, R., Kaufman, D., & Jones, S. (2018). Development and Evaluation of an Automated Machine Learning Algorithm for In-Hospital Mortality Risk Adjustment Among Critical Care Patients. *Critical Care Medicine*, *46*(6), e481–e488.
5. Parreco, J., Hidalgo, A., Badilla, A., Ilyas, O., & Rattan, R. (2018). Predicting central line-associated bloodstream infections and mortality using supervised machine learning. *Journal of Critical Care*, *45*, 156–162. https://doi.org/10.1016/j.jcrc.2018.02.010
6. Kim, S., Kim, S., Cho, J., Kim, Y., Sol, I., Sung, Y., Cho, I., Park, M., Jang, H., Kim, Y., Kim, K., & Sohn, M. (2019). A deep learning model for real-time mortality prediction in critically ill children. *Critical Care (London, England)*, *23*(1), 279–10. https://doi.org/10.1186/s13054-019-2561-z
7. Moll, M., Qiao, D., Regan, E., Hunninghake, G., Make, B., Tal-Singer, R., McGeachie, M., Castaldi, P., San Jose Estepar, R., Washko, G., Wells, J., LaFon, D., Strand, M., Bowler, R., Han, M., Vestbo, J., Celli, B., Calverley, P., Crapo, J., … Cho, M. (2020). Machine Learning and Prediction of All-Cause Mortality in Chronic Obstructive Pulmonary Disease. *Chest*.
8. Xu Z, Luo Y, Adekkanattu P, et al. Stratified Mortality Prediction of Patients with Acute Kidney Injury in Critical Care. *Stud Health Technol Inform*. 2019;264:462-466. doi:10.3233/SHTI190264
9. Ruiz, V., Saenz, L., Lopez-Magallon, A., Shields, A., Ogoe, H., Suresh, S., Munoz, R., & Tsui, F. (2019). Early prediction of critical events for infants with single-ventricle physiology in critical care using routinely collected data. *The Journal of Thoracic and Cardiovascular Surgery*, *158*(1), 234–243.e3. https://doi.org/10.1016/j.jtcvs.2019.01.130
10. Mayampurath, A., Sanchez-Pinto, L., Carey, K., Venable, L., & Churpek, M. (2019). Combining patient visual timelines with deep learning to predict mortality. *PloS One*, *14*(7), e0220640–. https://doi.org/10.1371/journal.pone.0220640
11. Lin, K., Hu, Y., & Kong, G. (2019). Predicting in-hospital mortality of patients with acute kidney injury in the ICU using random forest model. *International Journal of Medical Informatics (Shannon, Ireland)*, *125*, 55–61. https://doi.org/10.1016/j.ijmedinf.2019.02.002
12. Marafino, B., Park, M., Davies, J., Thombley, R., Luft, H., Sing, D., Kazi, D., DeJong, C., Boscardin, W., Dean, M., & Dudley, R. (2018). Validation of Prediction Models for Critical Care Outcomes Using Natural Language Processing of Electronic Health Record Data. *JAMA Network Open*, *1*(8), e185097–. https://doi.org/10.1001/jamanetworkopen.2018.5097
13. Pirracchio R, Petersen ML, Carone M, Rigon MR, Chevret S, van der Laan MJ. Mortality prediction in intensive care units with the Super ICU Learner Algorithm (SICULA): a population-based study. *Lancet Respir Med*. 2015;3(1):42-52. doi:10.1016/S2213-2600(14)70239-5
14. Anand, R., Stey, P., Jain, S., Biron, D., Bhatt, H., Monteiro, K., Feller, E., Ranney, M., Sarkar, I., & Chen, E. (2018). Predicting Mortality in Diabetic ICU Patients Using Machine

Learning and Severity Indices. *AMIA Summits on Translational Science Proceedings*, *2017*, 310–.
15. Aperstein, Y., Cohen, L., Bendavid, I., Cohen, J., Grozovsky, E., Rotem, T., & Singer, P. (2019). Improved ICU mortality prediction based on SOFA scores and gastrointestinal parameters. *PloS One*, *14*(9), e0222599–. https://doi.org/10.1371/journal.pone.0222599
16. Raj, R., Luostarinen, T., Pursiainen, E., Posti, J., Takala, R., Bendel, S., Konttila, T., & Korja, M. (2019). Machine learning-based dynamic mortality prediction after traumatic brain injury. *Scientific Reports*, *9*(1), 1–13. https://doi.org/10.1038/s41598-019-53889-6
17. Chiew CJ, Liu N, Wong TH, Sim YE, Abdullah HR. Utilizing Machine Learning Methods for Preoperative Prediction of Postsurgical Mortality and Intensive Care Unit Admission [published online ahead of print, 2019 Apr 8]. *Ann Surg*. 2019;10.1097/SLA.0000000000003297. doi:10.1097/SLA.0000000000003297
18. Hsieh, M., Hsieh, M., Chen, C., Hsieh, C., Chao, C., & Lai, C. (2018). Comparison of machine learning models for the prediction of mortality of patients with unplanned extubation in intensive care units. *Scientific Reports*, *8*(1), 17116–17117. https://doi.org/10.1038/s41598-018-35582-2
19. Nanayakkara, S., Fogarty, S., Tremeer, M., Ross, K., Richards, B., Bergmeir, C., Xu, S., Stub, D., Smith, K., Tacey, M., Liew, D., Pilcher, D., & Kaye, D. (2018). Characterising risk of in-hospital mortality following cardiac arrest using machine learning: A retrospective international registry study. *PLoS Medicine*, *15*(11), e1002709–. https://doi.org/10.1371/journal.pmed.1002709
20. Jain, S., Sarkar, I., Stey, P., Anand, R., Biron, D., & Chen, E. (2018). Using Demographic Factors and Comorbidities to Develop a Predictive Model for ICU Mortality in Patients with Acute Exacerbation COPD. *AMIA ... Annual Symposium Proceedings*, *2018*, 1319–1328.
21. Ge, W., Huh, J., Park, Y., Lee, J., Kim, Y., & Turchin, A. (2018). An Interpretable ICU Mortality Prediction Model Based on Logistic Regression and Recurrent Neural Networks with LSTM units. *AMIA ... Annual Symposium Proceedings*, *2018*, 460–469.
22. Sánchez Fernández, I., Sansevere, A., Gaínza-Lein, M., Kapur, K., & Loddenkemper, T. (2018). Machine Learning for Outcome Prediction in Electroencephalograph (EEG)-Monitored Children in the Intensive Care Unit. *Journal of Child Neurology*, *33*(8), 546–553. https://doi.org/10.1177/0883073818773230
23. Aushev, A., Ripoll, V., Vellido, A., Aletti, F., Pinto, B., Herpain, A., Post, E., Medina, E., Ferrer, R., Baselli, G., & Bendjelid, K. (2018). Feature selection for the accurate prediction of septic and cardiogenic shock ICU mortality in the acute phase. *PloS One*, *13*(11), e0199089–. https://doi.org/10.1371/journal.pone.0199089
24. Weissman, G., Hubbard, R., Ungar, L., Harhay, M., Greene, C., Himes, B., & Halpern, S. (2018). Inclusion of Unstructured Clinical Text Improves Early Prediction of Death or Prolonged ICU Stay. *Critical Care Medicine*, *46*(7), 1125–1132. https://doi.org/10.1097/ccm.0000000000003148
25. Hu CA, Chen CM, Fang YC, et al (2020). Using a machine learning approach to predict mortality in critically ill influenza patients: a cross-sectional retrospective multicentre study in Taiwan. *BMJ Open*. 2020;10(2):e033898. Published 2020 Feb 25. doi:10.1136/bmjopen-2019-033898
26. Houthooft, R., Ruyssinck, J., van der Herten, J., Stijven, S., Couckuyt, I., Gadeyne, B., Ongenae, F., Colpaert, K., Decruyenaere, J., Dhaene, T., & De Turck, F. (2015). Predictive modelling of survival and length of stay in critically ill patients using sequential organ failure scores. *Artificial Intelligence in Medicine*, *63*(3), 191–207. https://doi.org/10.1016/j.artmed.2014.12.009

27. Davoodi, R., & Moradi, M. (2018). Mortality prediction in intensive care units (ICUs) using a deep rule-based fuzzy classifier. *Journal of Biomedical Informatics*, *79*, 48–59. https://doi.org/10.1016/j.jbi.2018.02.008
28. Holmgren, G., Andersson, P., Jakobsson, A., & Frigyesi, A. (2019). Artificial neural networks improve and simplify intensive care mortality prognostication: a national cohort study of 217,289 first-time intensive care unit admissions. *Journal of Intensive Care*, *7*(1), 44–48. https://doi.org/10.1186/s40560-019-0393-1
29. Veith, N., & Steele, R. (2018). *Machine Learning-based Prediction of ICU Patient Mortality at Time of Admission*. 34–38. https://doi.org/10.1145/3206098.3206116
30. Krajnak, M., Xue, J., Kaiser, W., & Balloni, W. (2012). *Combining machine learning and clinical rules to build an algorithm for predicting ICU mortality risk*. 401–404.
31. Balkan B, Essay P, Subbian V. Evaluating ICU Clinical Severity Scoring Systems and Machine Learning Applications: APACHE IV/IVa Case Study. *Conf Proc IEEE Eng Med Biol Soc*. 2018;2018:4073-4076. doi:10.1109/EMBC.2018.8513324
32. Kang, M., Kim, J., Kim, D., Oh, K., Joo, K., Kim, Y., & Han, S. (2020). Machine learning algorithm to predict mortality in patients undergoing continuous renal replacement therapy. *Critical Care (London, England)*, *24*(1), 42–49. https://doi.org/10.1186/s13054-020-2752-7
33. Liu, J., Chen, X., Fang, L., Li, J., Yang, T., Zhan, Q., Tong, K., & Fang, Z. (2018). Mortality prediction based on imbalanced high-dimensional ICU big data. *Computers in Industry*, *98*, 218–225. https://doi.org/10.1016/j.compind.2018.01.017
34. Giannini, H., Ginestra, J., Chivers, C., Draugelis, M., Hanish, A., Schweickert, W., Fuchs, B., Meadows, L., Lynch, M., Donnelly, P., Pavan, K., Fishman, N., Hanson, C., & Umscheid, C. (2019). A Machine Learning Algorithm to Predict Severe Sepsis and Septic Shock: Development, Implementation, and Impact on Clinical Practice. *Critical Care Medicine*, *47*(11), 1485–1492. https://doi.org/10.1097/CCM.0000000000003891
35. Rojas, J., Carey, K., Edelson, D., Venable, L., Howell, M., & Churpek, M. (2018). Predicting Intensive Care Unit Readmission with Machine Learning Using Electronic Health Record Data. *Annals of the American Thoracic Society*, *15*(7), 846–853. https://doi.org/10.1513/AnnalsATS.201710-787OC
36. Johnson, A., Pollard, T., Shen, L. *et al.* MIMIC-III, a freely accessible critical care database. *Sci Data* **3,** 160035 (2016). https://doi.org/10.1038/sdata.2016.35
37. Choi, J., Jang, J., Lim, Y., Jang, J., Lee, G., Yang, H., Cho, J., & Hyun, S. (2018). Performance on the APACHE II, SAPS II, SOFA and the OHCA score of post-cardiac arrest patients treated with therapeutic hypothermia. *PloS One*, *13*(5), e0196197–. https://doi.org/10.1371/journal.pone.0196197
38. Falcão, A., Barros, A., Bezerra, A., Ferreira, N., Logato, C., Silva, F., do Monte, A., Tonella, R., de Figueiredo, L., Moreno, R., Dragosavac, D., & Andreollo, N. (2019). The prognostic accuracy evaluation of SAPS 3, SOFA and APACHE II scores for mortality prediction in the surgical ICU: an external validation study and decision-making analysis. *Annals of Intensive Care*, *9*(1), 1–10. https://doi.org/10.1186/s13613-019-0488-9
39. Ahmad, M., Eckert, C., & Teredesai, A. (2018). *Interpretable Machine Learning in Healthcare*. 559–560. https://doi.org/10.1145/3233547.3233667
40. Burrell, J. (2016). How the machine "thinks": Understanding opacity in machine learning algorithms. *Big Data & Society*, *3*(1), 205395171562251–.
41. Kuhn, M., Johnson, K. (2018). *Applied Predictive Modeling*. Springer.
42. James, G; Witten, D; Hastie, T; Tibshirani, R (2013). *An Introduction to Statistical Learning*. Springer.
43. Brownlee, J. (2018). *XGBoost with Python*.

Appendix

Data Dictionary

Category	Variable Name	Unit of Measure	Data Type	Description
identifier	encounter_id	None	integer	Unique identifier associated with a patient unit stay
identifier	hospital_id	None	integer	Unique identifier associated with a hospital
identifier	patient_id	None	integer	Unique identifier associated with a patient
demographic	hospital_death	None	binary	Whether the patient died during this hospitalization
demographic	age	Years	numeric	The age of the patient on unit admission
demographic	bmi	kilograms/metres^2	string	The body mass index of the person on unit admission
demographic	elective_surgery	None	binary	Whether the patient was admitted to the hospital for an elective surgical operation
demographic	ethnicity	None	string	The common national or cultural tradition which the person belongs to
demographic	gender	None	string	The genotypical sex of the patient
demographic	height	centimetres	numeric	The height of the person on unit admission
demographic	hospital_admit_source	None	string	The location of the patient prior to being admitted to the hospital
demographic	icu_admit_source	None	string	The location of the patient prior to being admitted to the unit
demographic	icu_admit_type	None	string	The type of unit admission for the patient
demographic	icu_id	None	integer	A unique identifier for the unit to which the patient was admitted
demographic	icu_stay_type	None	string	
demographic	icu_type	None	string	A classification which indicates the type of care the unit is capable of providing
demographic	pre_icu_los_days	Days	numeric	The length of stay of the patient between hospital admission and unit admission
demographic	readmission_status	None	binary	Whether the current unit stay is the second (or greater) stay at an ICU within the same hospitalization
demographic	weight	kilograms	numeric	The weight (body mass) of the person on unit admission
APACHE covariate	albumin_apache	g/L	numeric	The albumin concentration measured during the first 24 hours which results in the highest APACHE III score
APACHE covariate	apache_2_diagnosis	None	string	The APACHE II diagnosis for the ICU admission
APACHE covariate	apache_3j_diagnosis	None	string	The APACHE III-J sub-diagnosis code which best describes the reason for the ICU admission

Category	Variable Name	Unit of Measure	Data Type	Description
APACHE covariate	apache_post_operative	None	binary	The APACHE operative status; 1 for post-operative, 0 for non-operative
APACHE covariate	arf_apache	None	binary	Whether the patient had acute renal failure during the first 24 hours of their unit stay, defined as a 24 hour urine output <410ml, creatinine >=133 micromol/L and no chronic dialysis
APACHE covariate	bilirubin_apache	micromol/L	numeric	The bilirubin concentration measured during the first 24 hours which results in the highest APACHE III score
APACHE covariate	bun_apache	mmol/L	numeric	The blood urea nitrogen concentration measured during the first 24 hours which results in the highest APACHE III score
APACHE covariate	creatinine_apache	micromol/L	numeric	The creatinine concentration measured during the first 24 hours which results in the highest APACHE III score
APACHE covariate	fio2_apache	Fraction	numeric	The fraction of inspired oxygen from the arterial blood gas taken during the first 24 hours of unit admission which produces the highest APACHE III score for oxygenation
APACHE covariate	gcs_eyes_apache	None	integer	The eye opening component of the Glasgow Coma Scale measured during the first 24 hours which results in the highest APACHE III score
APACHE covariate	gcs_motor_apache	None	integer	The motor component of the Glasgow Coma Scale measured during the first 24 hours which results in the highest APACHE III score
APACHE covariate	gcs_unable_apache	None	binary	Whether the Glasgow Coma Scale was unable to be assessed due to patient sedation
APACHE covariate	gcs_verbal_apache	None	integer	The verbal component of the Glasgow Coma Scale measured during the first 24 hours which results in the highest APACHE III score
APACHE covariate	glucose_apache	mmol/L	numeric	The glucose concentration measured during the first 24 hours which results in the highest APACHE III score
APACHE covariate	heart_rate_apache	Beats per minute	numeric	The heart rate measured during the first 24 hours which results in the highest APACHE III score
APACHE covariate	hematocrit_apache	Fraction	numeric	The hematocrit measured during the first 24 hours which results in the highest APACHE III score
APACHE covariate	intubated_apache	None	binary	Whether the patient was intubated at the time of the highest scoring arterial blood gas used in the oxygenation score
APACHE covariate	map_apache	Millimetres of mercury	numeric	The mean arterial pressure measured during the first 24 hours which results in the highest APACHE III score
APACHE covariate	paco2_apache	Millimetres of mercury	numeric	The partial pressure of carbon dioxide from the arterial blood gas taken during the first 24 hours of unit admission which produces the highest APACHE III score for oxygenation
APACHE covariate	paco2_for_ph_apache	Millimetres of mercury	numeric	The partial pressure of carbon dioxide from the arterial blood gas taken during the first 24 hours

Category	Variable Name	Unit of Measure	Data Type	Description
				of unit admission which produces the highest APACHE III score for acid-base disturbance
APACHE covariate	pao2_apache	Millimetres of mercury	numeric	The partial pressure of oxygen from the arterial blood gas taken during the first 24 hours of unit admission which produces the highest APACHE III score for oxygenation
APACHE covariate	ph_apache	None	numeric	The pH from the arterial blood gas taken during the first 24 hours of unit admission which produces the highest APACHE III score for acid-base disturbance
APACHE covariate	resprate_apache	Breaths per minute	numeric	The respiratory rate measured during the first 24 hours which results in the highest APACHE III score
APACHE covariate	sodium_apache	mmol/L	numeric	The sodium concentration measured during the first 24 hours which results in the highest APACHE III score
APACHE covariate	temp_apache	Degrees Celsius	numeric	The temperature measured during the first 24 hours which results in the highest APACHE III score
APACHE covariate	urineoutput_apache	Millilitres	numeric	The total urine output for the first 24 hours
APACHE covariate	ventilated_apache	None	binary	Whether the patient was invasively ventilated at the time of the highest scoring arterial blood gas using the oxygenation scoring algorithm, including any mode of positive pressure ventilation delivered through a circuit attached to an endo-tracheal tube or tracheostomy
APACHE covariate	wbc_apache	10^9/L	numeric	The white blood cell count measured during the first 24 hours which results in the highest APACHE III score
vitals	d1_diasbp_invasive_max	Millimetres of mercury	numeric	The patient's highest diastolic blood pressure during the first 24 hours of their unit stay, invasively measured
vitals	d1_diasbp_invasive_min	Millimetres of mercury	numeric	The patient's lowest diastolic blood pressure during the first 24 hours of their unit stay, invasively measured
vitals	d1_diasbp_max	Millimetres of mercury	numeric	The patient's highest diastolic blood pressure during the first 24 hours of their unit stay, either non-invasively or invasively measured
vitals	d1_diasbp_min	Millimetres of mercury	numeric	The patient's lowest diastolic blood pressure during the first 24 hours of their unit stay, either non-invasively or invasively measured
vitals	d1_diasbp_noninvasive_max	Millimetres of mercury	numeric	The patient's highest diastolic blood pressure during the first 24 hours of their unit stay, non-invasively measured
vitals	d1_diasbp_noninvasive_min	Millimetres of mercury	numeric	The patient's lowest diastolic blood pressure during the first 24 hours of their unit stay, non-invasively measured
vitals	d1_heartrate_max	Beats per minute	numeric	The patient's highest heart rate during the first 24 hours of their unit stay

Category	Variable Name	Unit of Measure	Data Type	Description
vitals	d1_heartrate_min	Beats per minute	numeric	The patient's lowest heart rate during the first 24 hours of their unit stay
vitals	d1_mbp_invasive_max	Millimetres of mercury	numeric	The patient's highest mean blood pressure during the first 24 hours of their unit stay, invasively measured
vitals	d1_mbp_invasive_min	Millimetres of mercury	numeric	The patient's lowest mean blood pressure during the first 24 hours of their unit stay, invasively measured
vitals	d1_mbp_max	Millimetres of mercury	numeric	The patient's highest mean blood pressure during the first 24 hours of their unit stay, either non-invasively or invasively measured
vitals	d1_mbp_min	Millimetres of mercury	numeric	The patient's lowest mean blood pressure during the first 24 hours of their unit stay, either non-invasively or invasively measured
vitals	d1_mbp_noninvasive_max	Millimetres of mercury	numeric	The patient's highest mean blood pressure during the first 24 hours of their unit stay, non-invasively measured
vitals	d1_mbp_noninvasive_min	Millimetres of mercury	numeric	The patient's lowest mean blood pressure during the first 24 hours of their unit stay, non-invasively measured
vitals	d1_resprate_max	Breaths per minute	numeric	The patient's highest respiratory rate during the first 24 hours of their unit stay
vitals	d1_resprate_min	Breaths per minute	numeric	The patient's lowest respiratory rate during the first 24 hours of their unit stay
vitals	d1_spo2_max	Percentage	numeric	The patient's highest peripheral oxygen saturation during the first 24 hours of their unit stay
vitals	d1_spo2_min	Percentage	numeric	The patient's lowest peripheral oxygen saturation during the first 24 hours of their unit stay
vitals	d1_sysbp_invasive_max	Millimetres of mercury	numeric	The patient's highest systolic blood pressure during the first 24 hours of their unit stay, invasively measured
vitals	d1_sysbp_invasive_min	Millimetres of mercury	numeric	The patient's lowest systolic blood pressure during the first 24 hours of their unit stay, invasively measured
vitals	d1_sysbp_max	Millimetres of mercury	numeric	The patient's highest systolic blood pressure during the first 24 hours of their unit stay, either non-invasively or invasively measured
vitals	d1_sysbp_min	Millimetres of mercury	numeric	The patient's lowest systolic blood pressure during the first 24 hours of their unit stay, either non-invasively or invasively measured
vitals	d1_sysbp_noninvasive_max	Millimetres of mercury	numeric	The patient's highest systolic blood pressure during the first 24 hours of their unit stay, non-invasively measured
vitals	d1_sysbp_noninvasive_min	Millimetres of mercury	numeric	The patient's lowest systolic blood pressure during the first 24 hours of their unit stay, non-invasively measured
vitals	d1_temp_max	Degrees Celsius	numeric	The patient's highest core temperature during the first 24 hours of their unit stay, invasively measured

Category	Variable Name	Unit of Measure	Data Type	Description
vitals	d1_temp_min	Degrees Celsius	numeric	The patient's lowest core temperature during the first 24 hours of their unit stay
vitals	h1_diasbp_invasive_max	Millimetres of mercury	numeric	The patient's highest diastolic blood pressure during the first hour of their unit stay, invasively measured
vitals	h1_diasbp_invasive_min	Millimetres of mercury	numeric	The patient's lowest diastolic blood pressure during the first hour of their unit stay, invasively measured
vitals	h1_diasbp_max	Millimetres of mercury	numeric	The patient's highest diastolic blood pressure during the first hour of their unit stay, either non-invasively or invasively measured
vitals	h1_diasbp_min	Millimetres of mercury	numeric	The patient's lowest diastolic blood pressure during the first hour of their unit stay, either non-invasively or invasively measured
vitals	h1_diasbp_noninvasive_max	Millimetres of mercury	numeric	The patient's highest diastolic blood pressure during the first hour of their unit stay, non-invasively measured
vitals	h1_diasbp_noninvasive_min	Millimetres of mercury	numeric	The patient's lowest diastolic blood pressure during the first hour of their unit stay, non-invasively measured
vitals	h1_heartrate_max	Beats per minute	numeric	The patient's highest heart rate during the first hour of their unit stay
vitals	h1_heartrate_min	Beats per minute	numeric	The patient's lowest heart rate during the first hour of their unit stay
vitals	h1_mbp_invasive_max	Millimetres of mercury	numeric	The patient's highest mean blood pressure during the first hour of their unit stay, invasively measured
vitals	h1_mbp_invasive_min	Millimetres of mercury	numeric	The patient's lowest mean blood pressure during the first hour of their unit stay, invasively measured
vitals	h1_mbp_max	Millimetres of mercury	numeric	The patient's highest mean blood pressure during the first hour of their unit stay, either non-invasively or invasively measured
vitals	h1_mbp_min	Millimetres of mercury	numeric	The patient's lowest mean blood pressure during the first hour of their unit stay, either non-invasively or invasively measured
vitals	h1_mbp_noninvasive_max	Millimetres of mercury	numeric	The patient's highest mean blood pressure during the first hour of their unit stay, non-invasively measured
vitals	h1_mbp_noninvasive_min	Millimetres of mercury	numeric	The patient's lowest mean blood pressure during the first hour of their unit stay, non-invasively measured
vitals	h1_resprate_max	Breaths per minute	numeric	The patient's highest respiratory rate during the first hour of their unit stay
vitals	h1_resprate_min	Breaths per minute	numeric	The patient's lowest respiratory rate during the first hour of their unit stay
vitals	h1_spo2_max	Percentage	numeric	The patient's highest peripheral oxygen saturation during the first hour of their unit stay
vitals	h1_spo2_min	Percentage	numeric	The patient's lowest peripheral oxygen saturation during the first hour of their unit stay

Category	Variable Name	Unit of Measure	Data Type	Description
vitals	h1_sysbp_invasive_max	Millimetres of mercury	numeric	The patient's highest systolic blood pressure during the first hour of their unit stay, invasively measured
vitals	h1_sysbp_invasive_min	Millimetres of mercury	numeric	The patient's lowest systolic blood pressure during the first hour of their unit stay, invasively measured
vitals	h1_sysbp_max	Millimetres of mercury	numeric	The patient's highest systolic blood pressure during the first hour of their unit stay, either non-invasively or invasively measured
vitals	h1_sysbp_min	Millimetres of mercury	numeric	The patient's lowest systolic blood pressure during the first hour of their unit stay, either non-invasively or invasively measured
vitals	h1_sysbp_noninvasive_max	Millimetres of mercury	numeric	The patient's highest systolic blood pressure during the first hour of their unit stay, non-invasively measured
vitals	h1_sysbp_noninvasive_min	Millimetres of mercury	numeric	The patient's lowest systolic blood pressure during the first hour of their unit stay, non-invasively measured
vitals	h1_temp_max	Degrees Celsius	numeric	The patient's highest core temperature during the first hour of their unit stay, invasively measured
vitals	h1_temp_min	Degrees Celsius	numeric	The patient's lowest core temperature during the first hour of their unit stay
labs	d1_albumin_max	None	numeric	The lowest albumin concentration of the patient in their serum during the first 24 hours of their unit stay
labs	d1_albumin_min	g/L	numeric	The lowest albumin concentration of the patient in their serum during the first 24 hours of their unit stay
labs	d1_bilirubin_max	micromol/L	numeric	The highest bilirubin concentration of the patient in their serum or plasma during the first 24 hours of their unit stay
labs	d1_bilirubin_min	micromol/L	numeric	The lowest bilirubin concentration of the patient in their serum or plasma during the first 24 hours of their unit stay
labs	d1_bun_max	mmol/L	numeric	The highest blood urea nitrogen concentration of the patient in their serum or plasma during the first 24 hours of their unit stay
labs	d1_bun_min	mmol/L	numeric	The lowest blood urea nitrogen concentration of the patient in their serum or plasma during the first 24 hours of their unit stay
labs	d1_calcium_max	mmol/L	numeric	The highest calcium concentration of the patient in their serum during the first 24 hours of their unit stay
labs	d1_calcium_min	mmol/L	numeric	The lowest calcium concentration of the patient in their serum during the first 24 hours of their unit stay
labs	d1_creatinine_max	micromol/L	numeric	The highest creatinine concentration of the patient in their serum or plasma during the first 24 hours of their unit stay

Category	Variable Name	Unit of Measure	Data Type	Description
labs	d1_creatinine_min	micromol/L	numeric	The lowest creatinine concentration of the patient in their serum or plasma during the first 24 hours of their unit stay
labs	d1_glucose_max	mmol/L	numeric	The highest glucose concentration of the patient in their serum or plasma during the first 24 hours of their unit stay
labs	d1_glucose_min	mmol/L	numeric	The lowest glucose concentration of the patient in their serum or plasma during the first 24 hours of their unit stay
labs	d1_hco3_max	mmol/L	numeric	The highest bicarbonate concentration for the patient in their serum or plasma during the first 24 hours of their unit stay
labs	d1_hco3_min	None	numeric	The lowest bicarbonate concentration for the patient in their serum or plasma during the first 24 hours of their unit stay
labs	d1_hemaglobin_max	g/dL	numeric	The highest hemoglobin concentration for the patient during the first 24 hours of their unit stay
labs	d1_hemaglobin_min	g/dL	numeric	The lowest hemoglobin concentration for the patient during the first 24 hours of their unit stay
labs	d1_hematocrit_max	Fraction	numeric	The highest volume proportion of red blood cells in a patient's blood during the first 24 hours of their unit stay, expressed as a fraction
labs	d1_hematocrit_min	Fraction	numeric	The lowest volume proportion of red blood cells in a patient's blood during the first 24 hours of their unit stay, expressed as a fraction
labs	d1_inr_max	micromol/L	numeric	The highest international normalized ratio for the patient during the first 24 hours of their unit stay
labs	d1_inr_min	micromol/L	numeric	The lowest international normalized ratio for the patient during the first 24 hours of their unit stay
labs	d1_lactate_max	mmol/L	numeric	The highest lactate concentration for the patient in their serum or plasma during the first 24 hours of their unit stay
labs	d1_lactate_min	mmol/L	numeric	The lowest lactate concentration for the patient in their serum or plasma during the first 24 hours of their unit stay
labs	d1_platelets_max	10^9/L	numeric	The highest platelet count for the patient during the first 24 hours of their unit stay
labs	d1_platelets_min	10^9/L	numeric	The lowest platelet count for the patient during the first 24 hours of their unit stay
labs	d1_potassium_max	mmol/L	numeric	The highest potassium concentration for the patient in their serum or plasma during the first 24 hours of their unit stay
labs	d1_potassium_min	mmol/L	numeric	The lowest potassium concentration for the patient in their serum or plasma during the first 24 hours of their unit stay
labs	d1_sodium_max	mmol/L	numeric	The highest sodium concentration for the patient in their serum or plasma during the first 24 hours of their unit stay
labs	d1_sodium_min	mmol/L	numeric	The lowest sodium concentration for the patient in their serum or plasma during the first 24 hours of their unit stay

Category	Variable Name	Unit of Measure	Data Type	Description
labs	d1_wbc_max	10^9/L	numeric	The highest white blood cell count for the patient during the first 24 hours of their unit stay
labs	d1_wbc_min	10^9/L	numeric	The lowest white blood cell count for the patient during the first 24 hours of their unit stay
labs	h1_albumin_max	None	numeric	The lowest albumin concentration of the patient in their serum during the first hour of their unit stay
labs	h1_albumin_min	g/L	numeric	The lowest albumin concentration of the patient in their serum during the first hour of their unit stay
labs	h1_bilirubin_max	micromol/L	numeric	The highest bilirubin concentration of the patient in their serum or plasma during the first hour of their unit stay
labs	h1_bilirubin_min	micromol/L	numeric	The lowest bilirubin concentration of the patient in their serum or plasma during the first hour of their unit stay
labs	h1_bun_max	mmol/L	numeric	The highest blood urea nitrogen concentration of the patient in their serum or plasma during the first hour of their unit stay
labs	h1_bun_min	mmol/L	numeric	The lowest blood urea nitrogen concentration of the patient in their serum or plasma during the first hour of their unit stay
labs	h1_calcium_max	mmol/L	numeric	The highest calcium concentration of the patient in their serum during the first hour of their unit stay
labs	h1_calcium_min	mmol/L	numeric	The lowest calcium concentration of the patient in their serum during the first hour of their unit stay
labs	h1_creatinine_max	micromol/L	numeric	The highest creatinine concentration of the patient in their serum or plasma during the first hour of their unit stay
labs	h1_creatinine_min	micromol/L	numeric	The lowest creatinine concentration of the patient in their serum or plasma during the first hour of their unit stay
labs	h1_glucose_max	mmol/L	numeric	The highest glucose concentration of the patient in their serum or plasma during the first hour of their unit stay
labs	h1_glucose_min	mmol/L	numeric	The lowest glucose concentration of the patient in their serum or plasma during the first hour of their unit stay
labs	h1_hco3_max	mmol/L	numeric	The highest bicarbonate concentration for the patient in their serum or plasma during the first hour of their unit stay
labs	h1_hco3_min	None	numeric	The lowest bicarbonate concentration for the patient in their serum or plasma during the first hour of their unit stay
labs	h1_hemaglobin_max	g/dL	numeric	The highest hemoglobin concentration for the patient during the first hour of their unit stay
labs	h1_hemaglobin_min	g/dL	numeric	The lowest hemoglobin concentration for the patient during the first hour of their unit stay

Category	Variable Name	Unit of Measure	Data Type	Description
labs	h1_hematocrit_max	Fraction	numeric	The highest volume proportion of red blood cells in a patient's blood during the first hour of their unit stay, expressed as a fraction
labs	h1_hematocrit_min	Fraction	numeric	The lowest volume proportion of red blood cells in a patient's blood during the first hour of their unit stay, expressed as a fraction
labs	h1_inr_max	micromol/L	numeric	The highest international normalized ratio for the patient during the first hour of their unit stay
labs	h1_inr_min	micromol/L	numeric	The lowest international normalized ratio for the patient during the first hour of their unit stay
labs	h1_lactate_max	mmol/L	numeric	The highest lactate concentration for the patient in their serum or plasma during the first hour of their unit stay
labs	h1_lactate_min	mmol/L	numeric	The lowest lactate concentration for the patient in their serum or plasma during the first hour of their unit stay
labs	h1_platelets_max	10^9/L	numeric	The highest platelet count for the patient during the first hour of their unit stay
labs	h1_platelets_min	10^9/L	numeric	The lowest platelet count for the patient during the first hour of their unit stay
labs	h1_potassium_max	mmol/L	numeric	The highest potassium concentration for the patient in their serum or plasma during the first hour of their unit stay
labs	h1_potassium_min	mmol/L	numeric	The lowest potassium concentration for the patient in their serum or plasma during the first hour of their unit stay
labs	h1_sodium_max	mmol/L	numeric	The highest sodium concentration for the patient in their serum or plasma during the first hour of their unit stay
labs	h1_sodium_min	mmol/L	numeric	The lowest sodium concentration for the patient in their serum or plasma during the first hour of their unit stay
labs	h1_wbc_max	10^9/L	numeric	The highest white blood cell count for the patient during the first hour of their unit stay
labs	h1_wbc_min	10^9/L	numeric	The lowest white blood cell count for the patient during the first hour of their unit stay
labs blood gas	d1_arterial_pco2_max	Millimetres of mercury	numeric	The highest arterial partial pressure of carbon dioxide for the patient during the first 24 hours of their unit stay
labs blood gas	d1_arterial_pco2_min	Millimetres of mercury	numeric	The lowest arterial partial pressure of carbon dioxide for the patient during the first 24 hours of their unit stay
labs blood gas	d1_arterial_ph_max	None	numeric	The highest arterial pH for the patient during the first 24 hours of their unit stay
labs blood gas	d1_arterial_ph_min	None	numeric	The lowest arterial pH for the patient during the first 24 hours of their unit stay
labs blood gas	d1_arterial_po2_max	Millimetres of mercury	numeric	The highest arterial partial pressure of oxygen for the patient during the first 24 hours of their unit stay

Category	Variable Name	Unit of Measure	Data Type	Description
labs blood gas	d1_arterial_po2_min	Millimetres of mercury	numeric	The lowest arterial partial pressure of oxygen for the patient during the first 24 hours of their unit stay
labs blood gas	d1_pao2fio2ratio_max	Fraction	numeric	The highest fraction of inspired oxygen for the patient during the first 24 hours of their unit stay
labs blood gas	d1_pao2fio2ratio_min	Fraction	numeric	The lowest fraction of inspired oxygen for the patient during the first 24 hours of their unit stay
labs blood gas	h1_arterial_pco2_max	Millimetres of mercury	numeric	The highest arterial partial pressure of carbon dioxide for the patient during the first hour of their unit stay
labs blood gas	h1_arterial_pco2_min	Millimetres of mercury	numeric	The lowest arterial partial pressure of carbon dioxide for the patient during the first hour of their unit stay
labs blood gas	h1_arterial_ph_max	None	numeric	The highest arterial pH for the patient during the first hour of their unit stay
labs blood gas	h1_arterial_ph_min	None	numeric	The lowest arterial pH for the patient during the first hour of their unit stay
labs blood gas	h1_arterial_po2_max	Millimetres of mercury	numeric	The highest arterial partial pressure of oxygen for the patient during the first hour of their unit stay
labs blood gas	h1_arterial_po2_min	Millimetres of mercury	numeric	The lowest arterial partial pressure of oxygen for the patient during the first hour of their unit stay
labs blood gas	h1_pao2fio2ratio_max	Fraction	numeric	The highest fraction of inspired oxygen for the patient during the first hour of their unit stay
labs blood gas	h1_pao2fio2ratio_min	Fraction	numeric	The lowest fraction of inspired oxygen for the patient during the first hour of their unit stay
APACHE prediction	apache_4a_hospital_death_prob	None	numeric	The APACHE IVa probabilistic prediction of in-hospital mortality for the patient which utilizes the APACHE III score and other covariates, including diagnosis.
APACHE prediction	apache_4a_icu_death_prob	None	numeric	The APACHE IVa probabilistic prediction of in ICU mortality for the patient which utilizes the APACHE III score and other covariates, including diagnosis
APACHE comorbidity	aids	None	binary	Whether the patient has a definitive diagnosis of acquired immune deficiency syndrome (AIDS) (not HIV positive alone)
APACHE comorbidity	cirrhosis	None	binary	Whether the patient has a history of heavy alcohol use with portal hypertension and varices, other causes of cirrhosis with evidence of portal hypertension and varices, or biopsy proven cirrhosis. This comorbidity does not apply to patients with a functioning liver transplant.
APACHE comorbidity	diabetes_mellitus	None	binary	Whether the patient has been diagnosed with diabetes, either juvenile or adult onset, which requires medication.
APACHE comorbidity	hepatic_failure	None	binary	Whether the patient has cirrhosis and additional complications including jaundice and ascites, upper GI bleeding, hepatic encephalopathy, or coma.

Category	Variable Name	Unit of Measure	Data Type	Description
APACHE comorbidity	immunosuppression	None	binary	Whether the patient has their immune system suppressed within six months prior to ICU admission for any of the following reasons; radiation therapy, chemotherapy, use of non-cytotoxic immunosuppressive drugs, high dose steroids (at least 0.3 mg/kg/day of methylprednisolone or equivalent for at least 6 months).
APACHE comorbidity	leukemia	None	binary	Whether the patient has been diagnosed with acute or chronic myelogenous leukemia, acute or chronic lymphocytic leukemia, or multiple myeloma.
APACHE comorbidity	lymphoma	None	binary	Whether the patient has been diagnosed with non-Hodgkin lymphoma.
APACHE comorbidity	solid_tumor_with_metastasis	None	binary	Whether the patient has been diagnosed with any solid tumor carcinoma (including malignant melanoma) which has evidence of metastasis.
APACHE grouping	apache_3j_bodysystem	None	string	Admission diagnosis group for APACHE III
APACHE grouping	apache_2_bodysystem	None	string	Admission diagnosis group for APACHE II
GOSSIS example prediction	pred	None	numeric	Example mortality prediction, shared as a 'baseline' based on one of the GOSSIS algorithm development models.