Final Report

Interpretable Predictions of Mortality in ICU Patients

by

Shu Yu Tew • Yee Voon Lee

Contents

1 Introduction

The intensive care unit (ICU) is a crucial and important department for hospitals as it not only provides treatments for patients, but also determine the cost and performance of the hospital. There are many factors that complicates the process of identifying whether a patient should be admitted to a general ward or the ICU. Given this fact, without having some established standards of comparison, it will be hard for health care providers to evaluate and produce a meaningful report on the quality of patient care for each hospital. Comparisons made in such a case would be unfair at all times [5]. Hence it is in our best interest to build a model to predict ICU patients' mortality for 2 reasons: i) Aid clinicians to decide on the types of treatment or care process to be provided so that patients will benefit most from it, ii) Hospitals can optimally utilize health care resource. In the recent years, there has been an increasing focus on predicting mortality of ICU patients in the medical community due to the severity of ill patients and costly resources. Many had focus on implementing machine learning models to predict mortality based on the physiological data measured which provides valuable clinical data analysis [5].

This project aims to explore the performance of various machine learning algorithm in developing a mortality prediction model. This is achieved by analyzing patients' clinical records available from the Physio-Net/CinC challenge 2012 web page [2]. The data set includes records for 12,000 patients with explanatory variables inclusive of vital signs and pathological readings collected during the first 48 hours after admission into the ICU. The primary outcome of this project is to identify patients with high risk of death in their ICU stay, aside from developing a explainer model to interpret the reasoning behind this identification.

The objectives of this project are:

1. Build a predictive model that predicts binary outcomes, 0 being survival and 1 being in-hospital death, and
2. Produce an estimated probability of survival
3. Build an explainer model to help ease interpretability of the predictive model
4. Integrate (1) and (2) to a simple standalone program that produces the outcome when medical variables are keyed in. The program's interactivity and interface will not be our main priority.

2 Related work in mortality prediction for ICU patients

2.1 Scoring System

There has been a rapid development in creating more quantitative and statistical system to assess medical conditions and care required, without trading of the association with clinical measurements. This breakthrough in medical research is the scoring system. The scoring system produces a score which is a number that measures the disease severity. Scores are usually calculated by using the data collected from the first 24hr, 48hr and 72hr of ICU admission. The usage of this scores as an input variable for other machine learning models to predict ICU mortality has been widely introduced recently and will be discussed in the following section. Many scoring methods were developed to measure the severity of illness for ICU patients and provide refined methods for benchmarking ICU performance. The traditional standard scoring methods are APACHE, SAPS, SOFA, NEWS, and qSOFA. APACHE and SAPS assess ICU performance by obtaining the physiological variables within 24 hr after ICU admission and assess disease severity from patient's demographic[5]. These scoring methods are widely used by researchers to improve their model. Some apply the scoring system to their models by using the score outcomes as inputs to their model, whereas some use it as a benchmark to measure the model's performance.

2.2 Machine Learning techniques

Ever since the major breakthrough in ICU mortality prediction, the use of machine learning algorithm for in-hospital mortality prediction has been an open area for investigation. Logistic regression, support vector machine(SVMs), artificial neural network(ANNs), bayesian classifier and many more are a few popular machine learning algorithms used by many researchers.

In the CinC Challenge, (Luca Citi et al.)[20] used support vector machine algorithm (SVM) to classify patients into two groups, -1: survival, or +1: in-hospital death. The raw output of the SVM model is used together with a constant term as regressors (predicted variables) in a generalised linear model (GLM), with probit link to produce an estimated probability of a patient's in-hospital death and a binary outcome. (Luca Citi et al.) claims that SVM is robust enough to deal with datasets that contains abundant of redundancy or even uninformative variables, which allows users to obtain good performance even without the need of a variable selection process.

Research has shown that logistic regression is the top most popular ML algorithm, though in some cases have been outperformed by artificial neural networks (ANNs) [11, 16]. While other research [17, 14, 8, 10, 9] found that Decision Trees (DTs) and Support Vector Machines (SVMs) performed better than ANN and logistic regression. One sixth of the available papers on the CinC challenge show that participants widely use logistic regression to predict mortality. However only one of the participant team (S.Vairavan et al.) [20] made it into top 3 for first event (binary prediction) and none made it into top 5 for the second event (estimated probability). A noticeable fact is (Srinivasan Vairavan et al.)[20] did not merely used logistic regression to predict the outcome. The output of a statistical model which is the Hidden Markov Model was used as a feature to train the logistic regression. Among the top 5 scores for both events, two of the predictive models used Bayesian classifier. (Martin Macas et al.)[15] and (Alisair EW et al.)[13] both made it to top 5 scores in both events. However, both of them uses different classifiers whereby (Martin Macas et al.) uses linear Bayes classification, (Alistair Ew et al.) uses a Bayesian Ensemble method. This suggest that Bayesian classifier works well in predicting both binary outcome and estimated probability.

Hence, we can conclude that there are no explicit result on which model works best, as none of it invariably outperform each other[5]. In most cases, one single machine learning algorithm is not sufficient to accurately predict the outcome of patients mortality. It is important to thoroughly understand our dataset and the expected outcome so that we could make use of combinations of different models to produce a desirable result. Different models have different strengths based on each individual aspect, the population of interest,variables measured, and the outcome being tested [5]. The accuracy of the predicted outcome could be affected by many factors including the selection of explanatory variables and the sample dataset collected.

3 Framework for ICU mortality prediction

This section would be discussing about the general overview of this project in the process of studying in-hospital mortality prediction. Figure 1 shown below is a framework that was proposed in our previous project proposal with slight adjustments. Different colours are used to represent different stages of the project and to identify tasks within each stage. In the second stage, we can notice that there are arrows represented with dashed lines. This suggest that the process is optional because some models perform better without any normalization of the data (log transformation nor standardization).

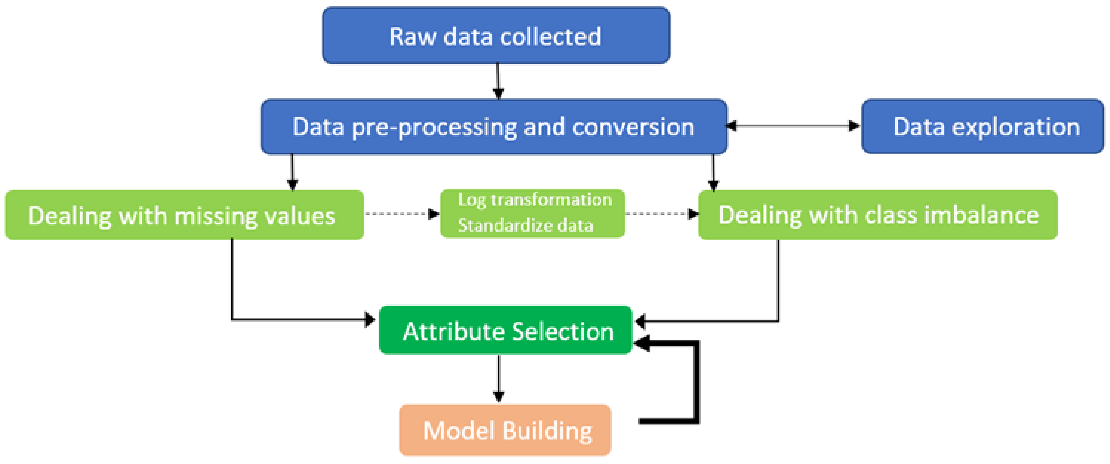


Figure 1: Project Framework

3.1 Data pre-processing and conversion

This stage involves compiling all 4,000 csv files into one final csv, imputing improbable values and feature creation. The 4,000 csv files comprise medical records for one patient. The final csv file would be having new variables created for each parameter. This process is repeated 3 times, each for a different datasets – set A, B and C. The outcome variables are also appended into the same csv file.

3.1.1 Cleaning improbable values

Our main issue dealing with ICU data is insufficient knowledge and understanding about all the variables provided. Therefore, it is hard for us to identify improbable values for all variables. Not all extreme values can be recognized as improbable because those values might be crucial in predicting one's death or survival. Therefore, this cleaning task is only done on logically obvious improbable values – for instance the negative values in the temperature variable. All 43 variables were supposed to be non-negative, however, we can observe that there are some observations in temperature that consists of negative values. These improbable negative values are treated by adding the negative values to the maximum record for that particular patient in the temperature variable. This applies to the weight variable as well. For weight less than 10kg, it is imputed with the summation of the value itself and the maximum weight recorded during first 48 hours after admission for that patient.

3.1.2 Feature creation

The ICUType variable consist of integers ranging from 1 to 4 each representing one different ICU unit. This impose an unnatural ordering [8]. Thus, three corresponding binary variables are created, for which if three of them is recorded as 0, it indicates that the record is from ICUType 1. A new variable "CumUrine" is introduced to record the cumulative sum of the Urine measurement.

This study is not about time series forecasting, it is more on using the time series records to solve a classification problem whereby the aim is to identify which set of categories a new patient belongs to – survival or in-hospital death. Therefore, we would need to create new variables that summaries the distribution and possible trends of each time series variables available for each patient.

The time series measurements were recorded at different point of time with different frequency of occurrence that vary across different patients. There are also different sets of variables recorded for each patient. Therefore, we decided to create binary variables for each of measurement to capture the absence of data in each patient. We then split the dataset into first 24 hours period and the last 24 hours period to retain some of the possible trend in the variables. This is because we would like to have a metric which evaluates the patient's condition between the first 24 hours and the next 24 hours, whether the condition of the patient is getting better or deteriorating. For each period, we computed the mean, standard deviation, minimum and maximum for each variable. This captures the average value of the measurements (mean), the variability of measurements between patients (min, max) and the variability of measurements of one patient within the 24 hours period (standard deviation).

3.2 Data Exploration

After compiling all patients' medical records into a single csv file with each parameter having their own column, we start to explore the distribution of the data. We plotted the distribution density for each variable to identify variables that requires transformation. For instance, we try to search for variables with skewed distribution such that log transformation could be performed. We created an Rscript that produces a pdf file with the distribution plots of all variables (distribution.R). From there, we scroll through all plots and identify variables that requires suitable transformation. This can be done because there is only 43 variables and therefore 43 plots to look through. It would be a hassle to manually look through all plots and identify the distribution of each variable if a dataset contains a huge amount of variables.

The sparseness of the dataset is explored by visualizing the percentage of missing values for each variable as shown in figure 2. Out of 43 variables, more than 30 variables are having more than 50% of their records missing. There are also a few variables having a percentage of absence close to 100% which are the first few variables plotted in figure 2. This gave us an overview on the dataset that we are dealing with and allow us to plan ahead on the imputation method that we would be implementing.

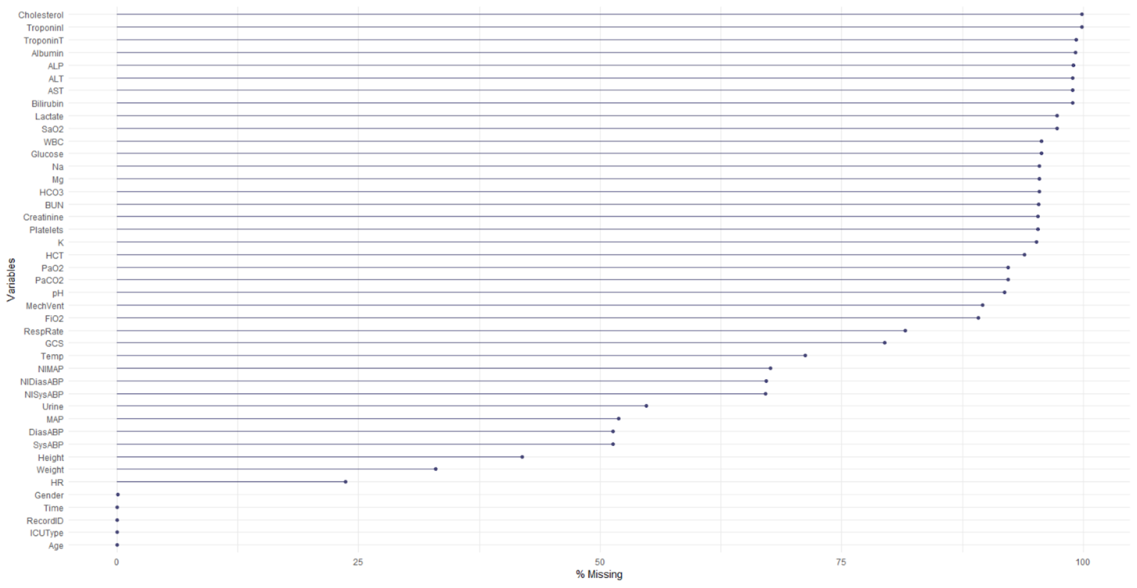


Figure 2: The percentage of missing values for each variable

We also looked at the range of values for each variable by plotting boxplots, as shown in figure 3. Boxplot provides a nice summary of the data showing the median and extreme values. This is how we discovered that there exist negative values for the temperature variable and those values need to be treated. Another realisation is that for variables like "Glucose" and "Platelet", the values range from 0 to more than 1000, whereas variables such as "Mg" has values only ranging from 0 to 10. This gave us an idea that the data might require standardization before training it on any model.

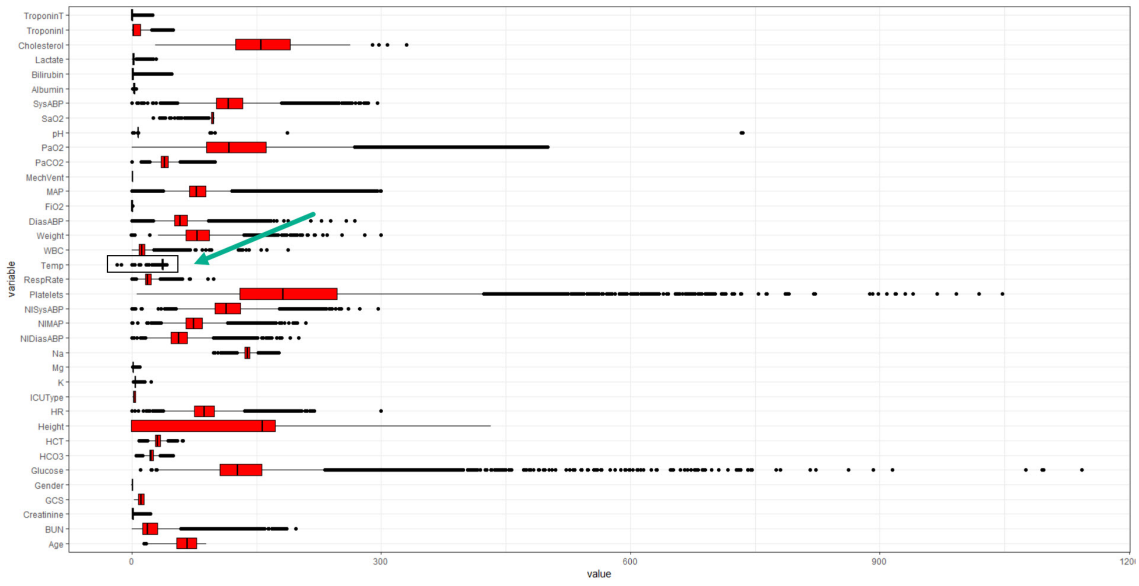


Figure 3: Boxplot of each variables

3.3 Normalization of variables

In general, classification works better if the features have comparable ranges and possibly, a Gaussian-like distribution [8]. Thus, for variables with skewed distribution such as any summary variables that is related to BUN (shown in figure 4), we would log transform the data first before re-scaling. Log transformation is not needed for variables such as HR (shown in figure 5) that is already normally distributed in nature. For such variables, we would jump straight to the re-scaling process. This process of deciding which variable requires transformation is done manually given that we only have 43 variables or less to work on. However, [8] suggested an automated procedure that attempts to apply appropriate transformation to each variable. From figure 3 we can notice that most variables consist the value zero. Taking the log of zero would return "-Inf", therefore we decided to log transform our data using the formula a log(x+1).

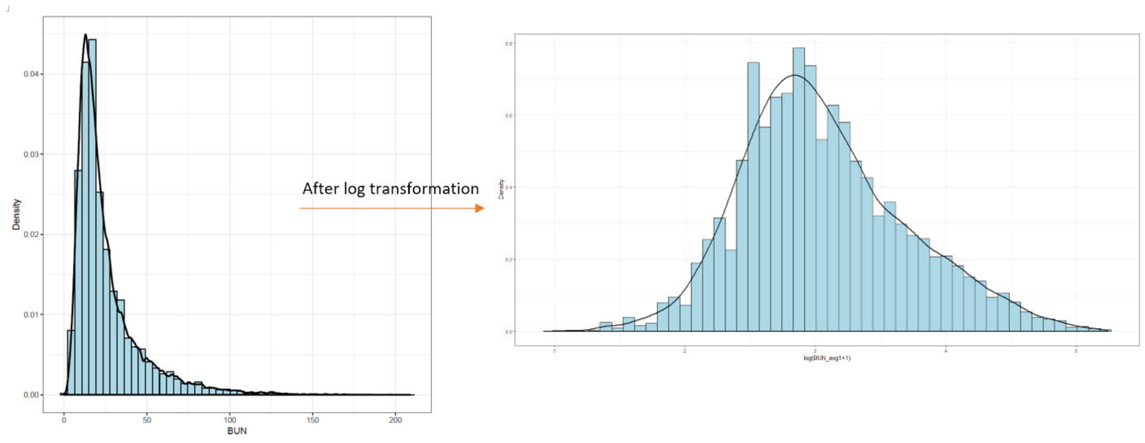


Figure 4: Distribution of "BUN" variable

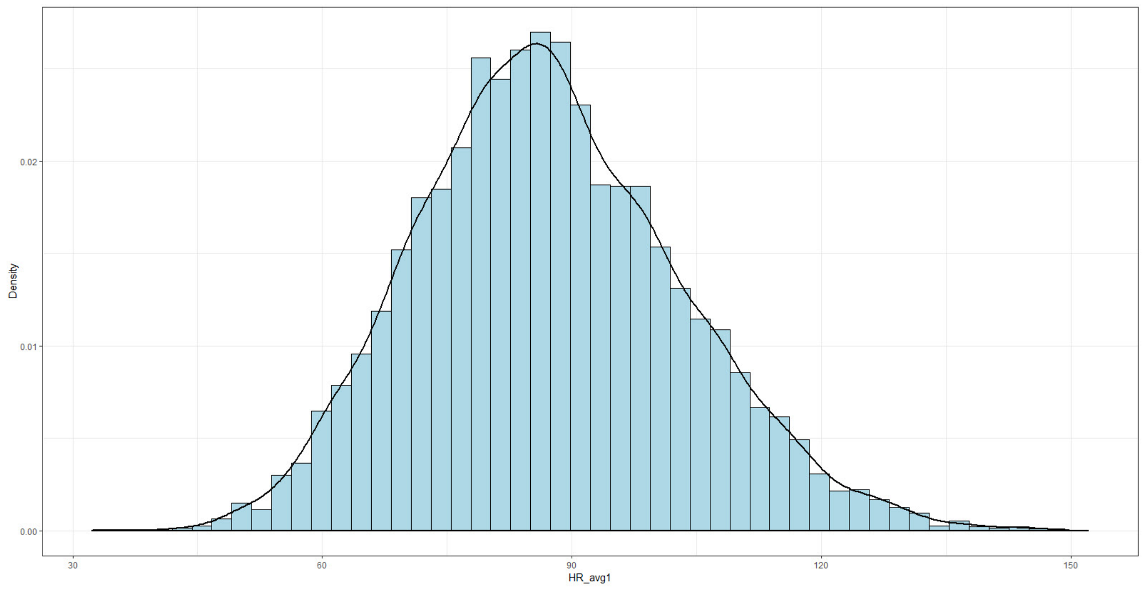


Figure 5: Distribution of "HR" variable

3.4 Missing values

Not all records for all variables are available for every patient. These incomplete data records are interpreted as a normal value whereby they are missing not at random (MNAR). Patients with multiple record entries missing may be due to the fact that they were regarded as being less sick than others, so they were not prioritized. Equally, the patient may have been regarded as being extremely sick to the extent that some screening or measurement process cannot be undertaken due to cases whereby the patient is too ill to even move. Therefore, it is crucial to understand the pattern of missing data to be able to identify the reason for data not being recorded.

Missing values can be treated either by excluding incomplete observation or by imputation. One of the most common technique would be to impute missing values with the mean or mode value of each attribute. It is impossible to remove incomplete observation in our study because there is at least more than one attribute with missing data for every patient. Therefore, instead of discarding variables which were missing for majority of the patients, we created binary variables for each attribute as discussed in Section 3.1.2

Two imputation method implemented in this project:

1. Mean – Mode imputation
   Impute mode for missing categorical data and impute mean for missing numerical values.
2. Predictive Mean Matching (PMM)
   Below is a brief description on how PMM works[1]. Suppose there is a single variable x that contains missing data and a set of variables z (with no missing data) that are used to impute x. Do the following:
   (a) For cases with no missing data, estimate a linear regression of x on z, producing a set of coefficients b.
   (b) Make a random draw from the "posterior predictive distribution" of b, producing a new set of coefficients b*. Typically this would be a random draw from a multivariate normal distribution with mean b and the estimated covariance matrix of b (with an additional random draw for the residual variance). This step is necessary to produce sufficient variability in the imputed values, and is common to all "proper" methods for multiple imputation.
   (c) Using b*, generate predicted values for x for all cases, both those with data missing on x and those with data present.
   (d) For each case with missing x, identify a set of cases with observed x whose predicted values are close to the predicted value for the case with missing data.
   (e) From among those close cases, randomly choose one and assign its observed value to substitute for the missing value.
   (f) Repeat steps 2 through 5 for each completed data set.

In this project, our set of variable z would be the descriptor variables and all the binary variables created at the data pre-processing stage that captures the frequency pattern of the data existence. PMM also helps to identify constant and collinear variables. Those variables will not be imputed and are removed from the training set.

3.5 Class Imbalance

Class imbalance is a major issue when working with ICU dataset because in nature, the number of patients with in-hospital death is relatively small as compared to the number of survivals. In this project, we will be using 2 different oversampling methods, the first method oversamples with replacement while the second method oversamples by creating synthetic data:

1. Random Over-Sampling Examples (ROSE)
   This method creates more samples by randomly oversampling minority samples.

2. Synthetic Minority Oversampling Technique (SMOTE)
   Generates sample using k-NN algorithm.

In this project, the sequence in which the data is pre-processed – i) imputation, (ii) log transform, (iii) rescaling and lastly, iv) oversampling. Such sequence is planned such that pre-processing (standardized) should be done before oversampling, since SMOTE uses a k-NN algorithm that applies Euclidean distance to generate samples. This requires data to be normalised to have better performance. If we were to use ROSE to oversample, it does not make much difference whether to standardize first or later because it is a random over-sampler.

3.6 Attribute selection

Halfway through modelling stage, we realised that the variable MechVent contain constant data whereby the records for every patient is all 1's. Therefore, we decided to remove this variable and use the binary variable created for MechVent instead, to identify the pattern in which when the variable is recorded.

To train our model, we used Random Forest, Decision Tree and Bayesian Generalize Linear Model to identify the top 10, 30 and 50 variables to test our scores on. We also removed the variables identified as constant and collinear by the PMM imputation method.

We selected 11 vital sign variables(including age) and 23 pathological variables as selected by [6] shown in table 1 to compare the models performance.

Table 1: Selected candidate variables

Vital Signs	Pathological
HR	Urine
(SysABP, NISysABP)	BUN, HCT, WBC, Glucose, K, Na, Mg,
(DiasABP, NIDiasABP)	HCO3, GCS, Creatinine, Albumin,
Temp	ALP, ALT, AST, Cholestrol, FiO2,
(RespRate, MechVent)	Lactate, PaO2, PaCO2, pH,
(NIMAP, MAP)	Platelets, SaO2

4 Project Management and Process

4.1 How project was conceived and managed

The project has been divided to several different parts with 4 deliverables and 3 milestones to achieve at different points of the project. This ensures that the project will be carried out smoothly and achieve the goal of the project. The 1st deliverable is the literature review which was completed in the first part of the project during the first semester. The literature review was done by researching and reviewing through many papers that discussed on similar topics. The papers published by the participants of the challenge and other related articles are used as the input for this project deliverable, which helped us through the process of learning and understanding the background, applications and past results of this topic.

With sufficient knowledge on this topic, we can move on to the 2nd project deliverable which is to develop prototype of a predictive model for patients' mortality. There are significant difference in types of admitted patients which directly affects both hospital and patient based variables and hence risk-adjustments are required to have an accurate and unbiased model. We used 2 different ways to achieve this deliverable; First, we run a given dataset through different models to observe the outcome. The next way is to run the same given dataset with same models but multiple iterations to also observe the outcome. We then made comparisons and decide when our predictions are successful enough to be developed into a prototype of a predictive model. That marks also the 1st milestone of the project, to develop a prototype of a predictive model to predict patient's mortality.

There are many great machine learning models that have achieved accurate predictions, but often times it could be too complex to interpret [7]. Besides that, we should never trust a model's outcome 100% without any explanations on how it is modelled. This leads us to our 3rd deliverable, to develop a prototype of an interpretable explainer model for our predictive model. There are many popular interpretation system that have been used by many researchers throughout the years. In this project, to ease the project work flow and ensure that we are on schedule, we have used LIME [18] interpretation system, which already has an existing R package, to develop the explainer model. Within a short time span, a local interpretable explainer model is then developed using LIME package in R. With that said, our 2nd milestone of the project is achieved.

The last part of the project is to deploy a program that predicts ICU mortality that comes with interpretable explanations. This is both the last deliverable and milestone of the project. This deliverable/milestone will be the final result of all the research and testing done throughout the entire project. The software will benefit experts in the medical field and also help hospitals improve their ICU performance in terms of quality of the medical services, quality and quantity of their facilities. This deliverable/milestone is achieved by using the Shiny package in R that builds interactive web apps. It is simple and straightforward to deploy our predictive models and explainer models to a standalone program layered with a simple UI.

4.2 Project development resources

1. ICU medical records released for the participants of PhysioNet CinC Challenge 2012
2. Software resource -
   (a) R programming language for modelling and deployment of program
   Open source packages and libraries: arm, caret, FNN, e1071, lime, smotefamily, randomForest, reshape2, rpart, rpart.plot, ROSE, rowr, varhandle, shiny, testthat, RUnit
   (b) Version controlling:
      i. Bitbucket for code development to ensure all team members have same version of code and monitor progress
      ii. Google drive for documentation
      iii. Overleaf for report documentations and referencing
3. Hardware resource - Team member's individual computing device, and additional computers provided by the university
4. Time and effort by team members to allocate to the project

4.3 Project planning or execution

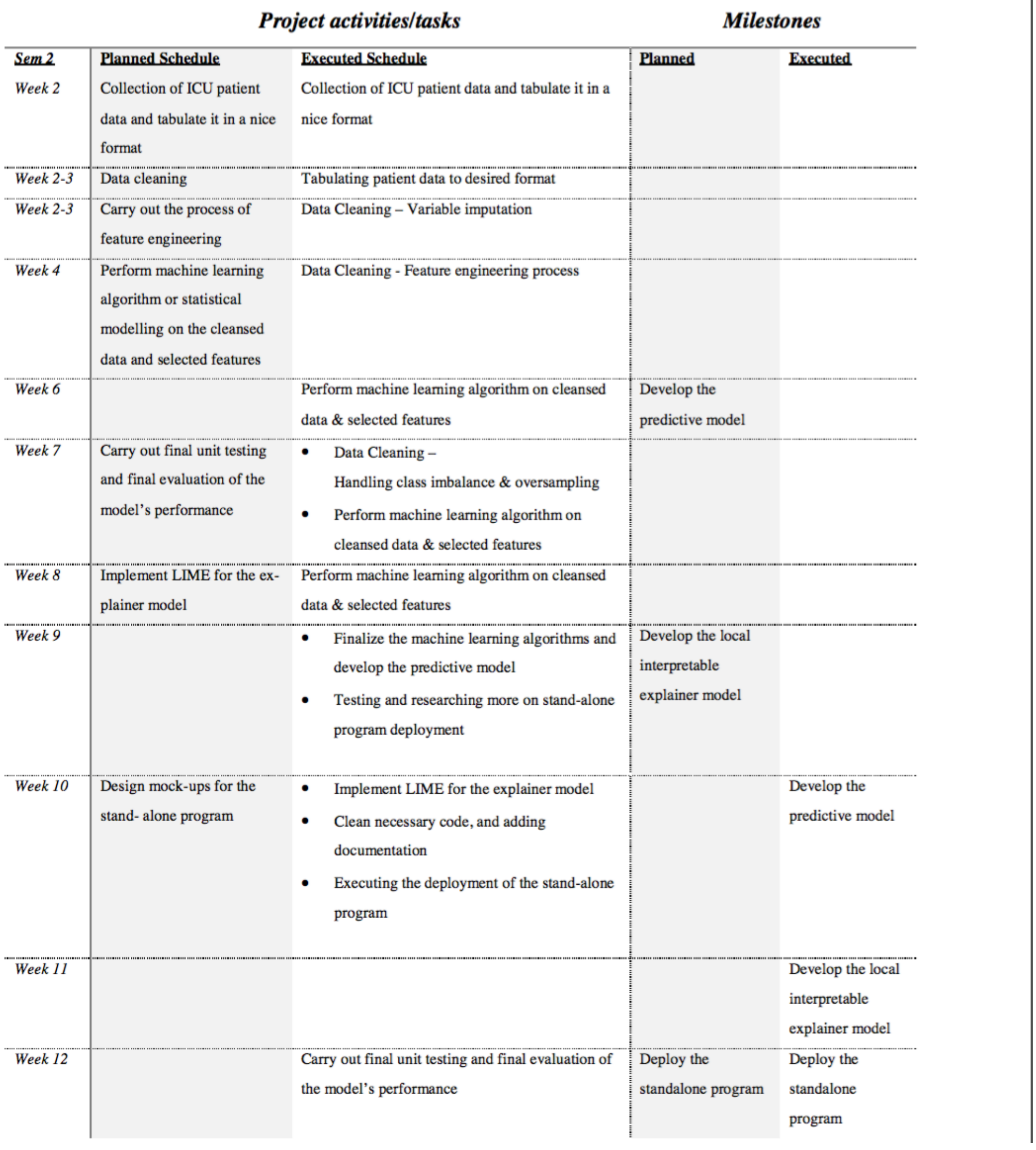
The execution of the project is managed by adapting to the Agile approach so that tasks can be refined and instead of delivering an outcome only when a specific stage is fully completed, agile helps deliver task frequently. It is also more flexible to changes and can easily adapt to it. Integrating agile process flow to the project:

1. Concept - To understand the importance and the nature of this project. Carry out as much research and studies on similar topics to familiar ourselves and grasp the concept, alongside gaining knowledge about this topic. Based on a ton of research, we have prioritized machine learning algorithms that produces comparatively better accuracy.
2. Iteration/Construction - Team members will be working on developing project deliverables, by carrying out feature selections, implementing machine learning algorithms, implementing explainer models and deploying a program
3. Release - QA testing and documentation development are carried out for every piece of code implemented. This process is iterated until satisfactory results are achieved, with review and improvements done at each iteration.

At the early stage, we have planned out in detail the schedule of the project. Throughout the project, there are many unforeseen circumstances that had caused delay in our tasks. For example, data preprocessing and cleaning took longer than expected. This is because during the beginning stage of performing selected machine learning algorithm, the model did not produce good accuracy, which leads us back to processing our data. Data preprocessing such as variable imputations, oversampling and feature engineering were carried out to help improve the accuracy of the model.

There were some delays at the deployment of the program stage as well. It was our initial plan to make use of Tensorflow for visualization and deployment for our program. However after various research, we realized that it was not very well suited for deployment of a standalone program. Throughout the research, we also came across Python Qt which we foresee would be a great and simple GUI toolkit as both of us are proficient in Python language. However we realized that integrating our predictive and explainer models which were coded in R would be slightly more complicated. Finally, we have decided to use R's existing package that is able to build a simple interactive standalone web app.

Table 6 below shows the planned schedule and the executed schedule for project tasks and milestones.



	Project activities/tasks		***Milestones***	
Sem 2	**Planned Schedule**	**Executed Schedule**	**Planned**	**Executed**
Week 2	Collection of ICU patient data and tabulate it in a nice format	Collection of ICU patient data and tabulate it in a nice format		
Week 2-3	Data cleaning	Tabulating patient data to desired format		
Week 2-3	Carry out the process of feature engineering	Data Cleaning – Variable imputation		
Week 4	Perform machine learning algorithm or statistical modelling on the cleansed data and selected features	Data Cleaning - Feature engineering process		
Week 6		Perform machine learning algorithm on cleansed data & selected features	Develop the predictive model	
Week 7	Carry out final unit testing and final evaluation of the model's performance	• Data Cleaning – Handling class imbalance & oversampling • Perform machine learning algorithm on cleansed data & selected features		
Week 8	Implement LIME for the explainer model	Perform machine learning algorithm on cleansed data & selected features		
Week 9		• Finalize the machine learning algorithms and develop the predictive model • Testing and researching more on stand-alone program deployment	Develop the local interpretable explainer model	
Week 10	Design mock-ups for the stand- alone program	• Implement LIME for the explainer model • Clean necessary code, and adding documentation • Executing the deployment of the stand-alone program		Develop the predictive model
Week 11				Develop the local interpretable explainer model
Week 12		Carry out final unit testing and final evaluation of the model's performance	Deploy the standalone program	Deploy the standalone program

Figure 6: Project's planned schedule vs the executed schedule

4.4 Risk Management

There are many way to identify potential risks in a project. The technique that we have used is by brainstorming all the potential underlying risks that our project could face. We have analyzed 4 risks in total for this project,

- Risk 1: due to limited dataset that is only from one region, it may cause our model to be inaccurate when tested with patient data from another region/country/hospital.
- Risk 2: Similar to Risk 1, the dataset given is very limited and it is complicated to retrieve more datasets as it involves authority issues. In the PhysioNet CinC challenge, participants were given 12,000 data in the dataset, and is able to split the data to 4,000 training data and 4,000 for testing data, and another 4,000 for a hidden test set. On the other hand, we only have 4,000 data to work with and split the data into training set and test set. Therefore, there might be a risk whereby our results are less accurate and too generalized as compared to the results from the participants of the competition.
- Risk 3: We will not only be using machine algorithms to carry out training and testing, but also use statistical models to model our data as well. These are 2 separate things, but have the same ultimate goal. However by using 2 different approach, it may lead to scope creep.
- Risk 4: Insufficient time to produce the deliverables at high quality. With other assignments and the given time frame of less than 6 months, we may have a risk of not having sufficient time to successfully deliver all project deliverables, or at least not at a high quality. To help manage some of the risks, we will be using the product backlog and our network diagram as a guide, so that we can possibly reduce the possibility of the risks. We have also set milestones to be achieved within certain period of time.

4.5 Limitations of the way the team worked

One of the biggest limitation the team had faced is version controlling, specifically syncing up codes for both team members. This could be because both of us are not very familiar with pair programming, so we are not quite familiar with how things should work. In pair programming, one person is the driver who writes code while the other is the observer who reviews each line of code as it is typed in. This could slowly become an issue when one of the pair potentially stop being actively engaged. The driver needs to "program aloud", as silently programming reduces the benefit. Another technical limitation faced is devices computation power is not strong enough that delays the run-time of the program. Teamwork also heavily relies on effective communication and providing constructive feedback to each other's work. With increasing amount of workload, it is sometimes difficult to effectively communicate with your partner and convey out the tasks that needs to be done. With that said, self-discipline from both team members are very important as well. While working individually forces one person to get work done, working in teams have the tendency to have a less effective work progress as the responsibility is divided by 2 people.

5 Mortality prediction using Machine Learning algorithms

This section presents the mortality prediction outcomes of top performing machine learning algorithms – Random Forest (RF), Decision Trees (DT), Bayesian Generalised Linear Model (bGLM) and Support Vector Machine (SVM). The score results are as shown in table 2.

Table 2: Predictive Model's outcome

Imputation	Over Sampling	Variables	RF	DT	SVM	bGLM
Mean	ROSE	All attributes	0.165 ± 0.023	0.323 ± 0.003	0.487 ± 0.010	0.388 ± 0.006
		Vital Signs	0.008 ± 0.009	0.248 ± 0.015	0.298 ± 0.008	0.281 ± 0.004
		Pathological Signs	0.191 ± 0.009	0.333 ± 0.011	0.428 ± 0.009	0.358 ± 0.018
		Vital + Pathological	0.172 ± 0.023	0.320 ± 0.022	0.464 ± 0.006	0.384 ± 0.011
		Top 15 attributes	0.322 ± 0.015	0.331 ± 0.022	-	0.356 ± 0.004
		Top 30 attributes	0.328 ± 0.008	-	-	0.358 ± 0.004
		Top 50 attributes	0.329 ± 0.015	-	-	0.364 ± 0.004
Mean	SMOTE	All attributes	0.357 ± 0.021	0.357 ± 0.006	0.439 ± 0.006	0.389 ± 0.003
		Vital Signs	0.270 ± 0.010	0.272 ± 0.013	0.276 ± 0.004	0.283 ± 0.003
		Pathological Signs	0.309 ± 0.011	0.342 ± 0.008	0.429 ± 0.004	0.375 ± 0.008
		Vital + Pathological	0.319 ± 0.028	0.358 ± 0.006	0.437 ± 0.009	0.387 ± 0.006
		Top 15 attributes	0.438 ± 0.011	0.408 ± 0.005	-	0.339 ± 0.008
		Top 30 attributes	0.426 ± 0.012	-	-	0.363 ± 0.002
		Top 50 attributes	0.436 ± 0.021	-	-	0.372 ± 0.003
Mice pmm	SMOTE	All attributes	0.427 ± 0.016	0.361 ± 0.006	0.401 ± 0.005	0.315 ± 0.008
		Vital Signs	0.273 ± 0.019	0.243 ± 0.000	0.283 ± 0.004	0.268 ± 0.003
		Pathological Signs	0.425 ± 0.011	0.345 ± 0.003	0.463 ± 0.006	0.345 ± 0.006
		Vital + Pathological	0.396 ± 0.019	0.362 ± 0.002	0.412 ± 0.009	0.340 ± 0.005
		Top 15 attributes	0.454 ± 0.011	0.336 ± 0.016	-	0.232 ± 0.002
		Top 30 attributes	0.442 ± 0.008	-	-	0.335 ± 0.004
		Top 50 attributes	0.459 ± 0.019	-	-	0.330 ± 0.005

We trained various model using 7 different sets of variables as listed in the "Variables" column of table 2. Data in set A is combined with data set B for models to train on. The performance of each model is then tested using data recorded in Set C. Every score comes with an error term for which it specifies the uncertainty of the score for that model. The uncertainty value is calculated after 10 iterations of modelling by using the below equation:

$$\frac{max(score) - min(score)}{2}$$

where max(score) = maximum score achieved within the 10 iteration, and
min(score) = minimum score achieved within the 10 iteration

Such approach is taken due to the fact that some machine learning algorithm involves randomization which would result in different score at different runs. The oversampling techniques used in this project is also random. Therefore, measures have to be taken to ensure reproducibility of the scores. This also provide us with a metric on how accurate our model is on average at predicting patient's in-hospital death instead of concluding the performance of an ML algorithm based on only one score.

5.1 Random Forest

Random Forest uses bootstrap sampling to build many different decision trees on the same dataset, whereby it involves a lot of randomization. Given that the oversampling methods also involves randomization, we decided to set seed when oversampling and only account for the randomization process done by the random forest algorithm. We would like to know how robust and stable Random forest algorithm is in mortality prediction using the same set of oversampled data. We were hoping that using the same dataset, the uncertainty of the scores would be lower. However, as observed in table 3, the uncertainty of scores obtain using random forest models are generally higher relative to the 3 other machine learning algorithm models. The most stable model with low uncertainty would be the Bayesian generalised linear model.

The above process to set seed when oversampling only applies for Random Forest because other algorithm used in this study is not random.

The result scores improved drastically when SMOTE oversampling technique is applied onto the dataset table 2. However, the performance of this model went down when the data is normalised. The big difference in score can be observed especially when the model is train only using the top 10, 30 and 50 variables. As such, we would also expect that the decision tree models would perform badly when the data is normalised. The reason for such phenomena to occur is discuss in the decision tree section 5.2.

Table 3: RF's outcome

Imputation	Over Sampling	Variables	Not Normalized	Normalized
Mean	SMOTE	All attributes	0.357 ± 0.021	
		Vital Signs	0.270 ± 0.010	
		Pathological Signs	0.309 ± 0.011	
		Vital + Pathological	0.319 ± 0.028	
		Top 15 attributes	0.438 ± 0.011	0.149 ± 0.006
		Top 30 attributes	0.426 ± 0.012	0.170 ± 0.008
		Top 50 attributes	0.436 ± 0.021	0.196 ± 0.014
Mice pmm	SMOTE	All attributes	0.427 ± 0.016	0.361 ± 0.018
		Vital Signs	0.273 ± 0.019	0.282 ± 0.011
		Pathological Signs	0.425 ± 0.011	0.311 ± 0.015
		Vital + Pathological	0.396 ± 0.019	0.341 ± 0.016
		Top 15 attributes	0.454 ± 0.011	0.155 ± 0.004
		Top 30 attributes	0.442 ± 0.008	0.186 ± 0.006
		Top 50 attributes	0.459 ± 0.019	0.214 ± 0.010

5.2 Decision Tree

The model produce uses less than 25 variables as nodes to split on, therefore, we were only able to test on the top 15 variables to compare the scores on. Decision tree performs the best when the data is neither log transform nor standardized. The reasoning might be that when the decision tree split between 2 values, it used the midpoint as split. If we log transform or standardize the data, that changes what a midpoint is. The midpoint between 2 points in the original space is no longer just halfway between them, but it is biased towards higher values[4].

It is widely believed that the prediction accuracy of decision trees models is invariant under any strict monotone transformation of individual predictor variables [12]. This fact has been said to be false when predicting new observations with values that were not seen in the training set and when they are close to the location of the split point of the tree[12]. With that being said, this suggest that our test set might be consisting patterns of data that's not observed in the training set. The score shown in table 2 is the score obtain when decision tree is trained on the original dataset (without log transformation nor standardization). If we were to train our decision tree on standardize data, the average scores obtained would be around 0.14.

The uncertainty of the scores are also generally lower than random forest despite the fact that the scores are slightly lower given that both algorithms are based on tree algorithm. This is because the random forest in our project is train by building 100 different trees for 10 iterations while decision tree is just training on one tree per iteration.

5.3 Support Vector Machine (SVM)

SVM is a discriminative classifier which finds a hyperplane in N-dimensional space (N – the number of features) that distinctly classifies data points [3]. This method produces high scores for each of the different subsets of dataset, in fact, it has the highest score of 0.48 when trained on mean imputed- Rose oversampled – using all attribute dataset. SVM performs better when the data is log transform and standardised. It also has an overall lower uncertainty as compared to random forest and decision tree.

5.4 Bayesian Generalise Linear Model (bGlm)

This method is similar to a logistic regression that has weakly informative priors for its logistics regression coefficient. The uncertainty of the scores obtain by this model is the lowest among all 4 models used in this project. This suggest that the performance of a bGlm model would be constantly good with an overall average score of 0.35++ regardless of the random sampling of datatset. Standardizing and log transformation of the data does not help much in improving the scores as the bayesglm function itself has a parameter "scaled" which by default is set to true. This suggest that the algorithm itself has already rescale the data for us.

5.5 Results and discussion

When comparing the performance of all experiments, we find that the model's prediction performance is better when using attributes that includes pathological signs. In general, SMOTE oversampling technique helps to improve the performance score all models. Although predictive mean matching imputation method produces a slightly higher average score for many models as compared to mean-mode imputation, the increase in classification performance is not significant given that pmm imputation takes more than 45 mins to run on a normal computer.
The scoring system using in this project is similar to of which used in the CinC challenge such that the scores obtained by the participant in the challenge can be used as benchmark to evaluate our model performances. It can also be observed that for some algorithms, training models on all variables does not necessarily give the best score results. We can see that when training models on random forest, the score is on average the highest when the model is trained on top 15 attributes. This suggest that feature selection is important in tuning our model performance.

6 LIME Explainer Model

Local Interpretable Model-agnostic Explanation (LIME) is used to identify an interpretable model over the interpretable representation that is locally faithful to our classifier[19]. From table 2, SVM is identified as the model that produces the highest score in mortality prediction when it is tested upon set c data. However, the dataset used in getting that highest score is the full original dataset that includes all variables. This will take very long for the model to be trained on before passing it into the explainer model given the fact that SVM also has a long run-time. Therefore, we decide to use only the top 50 variables identified by pmm-imputed-smote-oversampled-random forest model. This compromise is made to reduce the run-time of the code while retaining the high score of the predictive model. Although it is stated in section 5.5 that the scores are generally higher when pathological sign variables are included, it is not reasonable for us to include all pathological sign variables into our training set because we would need to include more than 200 variables to train on. Using the top 50 variables identified by the random forest model, we still get a very high score of 0.401 ± 0.0006. This will not affect the result of our explainer model because there's only so much variables that a doctor can go through within limited time. Therefore, only a limited number of variables that our explainer model needs to display in order to explain the prediction of our black box classifier (in this case is our support vector machine model).

For each patient, there would be a unique set of explanatory variables to explain the prediction made for that patient. The graph produced by the explainer model clearly indicates which variable is supporting the prediction of the survival or death of the patient, shown in figure 7.

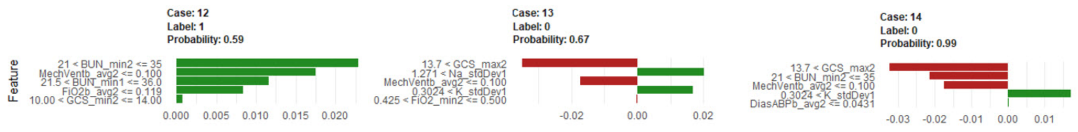


Figure 7: LIME Explainer model output

Figure 7 is the example output of our LIME explainer model. Looking at case 12, the explainer model predict patient 12 to be dead (1: dead, 0 survival). The top 4 variables that support that results are BUN_min2, MechVentb_avg2, BUN_min1, FiO2b_avg2 and GCS_min2. There is no display of variables that contradicts the outcome. The visual suggest the model predict death for patient 12 because the MechVentb_avg2 value is less than 0.1, the lesser it is, there is higher chance for the model to predict the patient as in-hospital death. Whereas for patient 14, the model predicts a 99% survival. The plot suggests that the higher the GCS readings for patient 14, there is a higher chance of survival. This make sense because in patient 12, the plot suggests that the lower the GCS values, the higher chance for in-hospital death of the patient.

7 Conclusion

There already exist many papers and researches on ICU mortality prediction. However, most of these studies are only designed to predict mortality without having an explainer model to explain the prediction of the black box classifier. Explanation of the outcome is needed for doctors to make decision on whether the prediction is reasonable in a medical context and to determine whether is the output trustworthy. This paper presented a general framework of mortality prediction with details on the process of building a model on mortality prediction – from how we dealt with the raw data to the measure taken in building our explainer model. This framework has been evaluated on ICU medical records of 8,000 patients extracted from the CinC challenge using several different machine learning algorithms. The results produced by participants of challenge are also used as benchmark to ensure that our output results are comparable.

Several important findings in our study:

1. The test set of our data might not be representative of our training set given that the random forest and decision tree scores went down drastically when the data is normalised.
2. SMOTE oversampling technique improves the classification performance of most models.
3. Models with high scores are produced when pathological sign variables are included to train the model.
4. Vital signs do not contribute much in predicting ICU patient's mortality as compared to using pathological sign variables.
5. Model performance can be improved after feature selection, not only by including all possible variables available.

One of the main drawbacks of this project is to not have a metric to evaluate the accuracy of the predict survival probability of each patients. This can be done by using the resources given in the CinC challenge webpage [2]. All in all, it is in our intention to build on this project and further explore various ways to improve the classification accuracy of our model aside from refining the explainer model.

References

[1] Imputation by predictive mean matching: Promise and peril.

[2] Predicting mortality of icu patients: the physionet/computing in cardiology challenge 2012.

[3] Support vector machine[U+200A]—[U+200A]introduction to machine learning algorithms.

[4] Why do data transformations affect performance of tree-based models?

[5] Aya Awad, Mohamed Bader-El-Den, James McNicholas, and Jim Briggs. Early hospital mortality prediction of intensive care unit patients using an ensemble learning approach. *International journal of medical informatics*, 108:185–195, 2017.

[6] Deep Bera and Mithun Manjnath Nayak. Mortality risk assessment for icu patients using logistic regression. In *Computing in Cardiology (CinC), 2012*, pages 493–496. IEEE, 2012.

[7] Dr. Christoph Bergmeir. Interpretable predictions of mortality in icu patients. 2018.

[8] Luca Citi and Riccardo Barbieri. Physionet 2012 challenge: Predicting mortality of icu patients using a cascaded svm-glm paradigm. In *Computing in Cardiology (CinC), 2012*, pages 257–260. IEEE, 2012.

[9] E David Crawford, Joseph T Batuello, Peter Snow, Eduard J Gamito, David G McLeod, Alan W Partin, Nelson Stone, James Montie, Richard Stock, John Lynch, et al. The use of artificial intelligence technology to predict lymph node spread in men with clinically localized prostate carcinoma. *Cancer: Interdisciplinary International Journal of the American Cancer Society*, 88(9):2105–2109, 2000.

[10] Dursun Delen, Glenn Walker, and Amit Kadam. Predicting breast cancer survivability: a comparison of three data mining methods. *Artificial intelligence in medicine*, 34(2):113–127, 2005.

[11] Richard Dybowski, V Gant, P Weller, and R Chang. Prediction of outcome in critically ill patients using artificial neural network synthesised by genetic algorithm. *The Lancet*, 347(9009):1146–1150, 1996.

[12] Tal Galili and Isaac Meilijson. Splitting matters: how monotone transformation of predictor variables may improve the predictions of decision tree models. *arXiv preprint arXiv:1611.04561*, 2016.

[13] Alistair EW Johnson, Nic Dunkley, Louis Mayaud, Athanasios Tsanas, Andrew A Kramer, and Gari D Clifford. Patient specific predictions in the intensive care unit using a bayesian ensemble. In *Computing in Cardiology (CinC), 2012*, pages 249–252. IEEE, 2012.

[14] Sujin Kim, Woojae Kim, and Rae Woong Park. A comparison of intensive care unit mortality prediction models through the use of data mining techniques. *Healthcare informatics research*, 17(4):232–243, 2011.

[15] Martin Macaš, Jakub Kuzilek, Tadeáš Odstrčilík, and Michal Huptych. Linear bayes classification for mortality prediction. In *Computing in Cardiology (CinC), 2012*, pages 473–476. IEEE, 2012.

[16] A Nimgaonkar and S Sudarshan. Predicting hospital mortality for patients in the intensive care unit: a comparison of artificial neural networks with logistic regression models. *Intensive Care Med*, 30:248–253, 2004.

[17] Vicent J Ribas, Jesús Caballero López, Adolf Ruiz-Sanmartín, Juan Carlos Ruiz-Rodríguez, Jordi Rello, Anna Wojdel, and Alfredo Vellido. Severe sepsis mortality prediction with relevance vector machines. In *Engineering in medicine and biology society, EMBC, 2011 annual international conference of the IEEE*, pages 100–103. IEEE, 2011.

[18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

[19] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

[20] Srinivasan Vairavan, Larry Eshelman, Syed Haider, Abigail Flower, and Adam Seiver. Prediction of mortality in an intensive care unit using logistic regression and a hidden markov model. *Computing in cardiology*, 39:393–396, 2012.