MONASH University

Final Project Report

Daniel Clark, Li Yunhao & Harry Lawson

Date: 21st of October 2022

Word count: 7005

Introduction

Intensive Care Units (ICU) provide care to critically ill patients who need close monitoring from doctors and nurses in order to survive. It is vital that hospitals, especially those with ICUs, assess and report on patient care to compare efficacy of interventions, such as the performance of a procedure or the administration of a medication. Mortality is a key metric used to determine the performance of a specific intensive care unit. However, an important issue for predicting the performance of hospital ICU performance is that hospitals take in patients with varying levels of mortality risk depending on their specialisation. Therefore, there must be a risk adjustment when comparing the performance of each hospital. To counteract this, a predictive model that uses all the data across different hospitals to predict mortality rates can be used as a benchmark for hospitals and ICUs to assess performance. In the current hospital space, simple predictive models are used to analyse ICU performance. An example of an industry standard model is the Acute Physiology Chronic Health Evaluation (APACHE), which is a highly interpretable model. Recently, it has been demonstrated that modern Machine Learning models can achieve superior predictions in this space. However, these methods have low interpretability and do not inform hospitals on what practical changes can be made to improve patient care in ICU. Therefore, there is an opportunity to create a new model that accurately predicts mortality in ICUs while retaining interpretable results that hospitals can use to make important procedural changes. Our predictive model has been trained by the dataset used for the 2012 PhysioNet Computing in Cardiology Challenge that contains 12,000 ICU patient stays. In this challenge, various data pre-processing techniques and advanced machine learning algorithms were used to predict mortality. Since 2012, machine learning algorithms and interpretable explainer models have made significant advancements. Thus, using the findings from the PhysioNet challenges and the advancements in machine learning, our project has made significant improvements in predicting and interpreting mortality in ICUs. Furthermore, our program can be accessed via a web browser on devices available to hospital's and the program is operable for hospital workers with beginner level computing skills. The combination of an improved predictive model and our end user friendly software can benefit hospitals in improving patient care in ICU.

The following project report details what has been implemented in our project and how our project requirements were met. Furthermore, it justifies key decisions made regarding the final web application, the limitations of these decisions. and discusses possible improvements in the future. The methodology used for this project is provided in this report and includes Python, Flask, Tensorflow, Sklearn, HTML, CSS, Pandas, and SQLAlchemy. The software deliverables are discussed, and evidence of these deliverables are included in the final appendix. Finally, the report includes a critical discussion of the outcomes of our project before its conclusion.

Project Background

Brief background introduction

Not all patients in ICU settings are going to survive. It is necessary that medical personnel try their best to save as many lives as possible, however there are many complicated factors for every individual being treated. There is more to consider when it comes to the state of patients and what changes in their condition suggest than can always be manually processed by medical staff. Providing medical staff with automated analysis and potentially improved analysis could potentially allow them to save more lives, which is why our project is focused on producing a machine learning model for ICU mortality predictions. Previously, some traditional methods assisted health professionals in making predictions, however these models are no longer necessarily the best service we can utilise. Our project involves producing a machine learning model which will process time-series data from various patients along with their mortality status, and will provide hospitals with an indication of how many patients survived compared to how many patients the model predicted would survive. The application allows hospital staff to view specific patients and see what factors contributed to their survival likelihood, so it does not directly assist in prevention. The service allows hospitals to assess their previous performance in order to improve their services in the future.

Literature Review

Introduction

Intensive Care Unit (ICU) settings are the medical facilities where the most critical medical care takes place. The proportion of patients in ICU that die during their stay is significantly higher than in most other medical settings. To reduce the probability of death during an ICU stay, we want to be able to improve predictive methods. The faster and more accurate the technique, the more chance that we can appropriately treat the most critically ill patients effectively. For decades, weighted traditional methods such as Sequential Organ Failure Assessment (SOFA), Acute Physiology and Chronic Health Evaluation (APACHE) and many other general and specific measures have been used as an assessment tool for risk. However, machine learning techniques have recently shown promise for future risk assessment. The 2012 PhysioNet challenge (Silva et al. 2012), which involved 37 teams worldwide, was a computing challenge with the intention to improve analysis techniques and to produce improved predictor techniques for ICU mortality. For the purposes of this review, the concept of accuracy will refer to both how high the successful death prediction rate is and how low the false death prediction rate is.

Contemporary machine learning implementations have since been employed, which can allow us to not only predict risk of death but also produce interpretable explanations of the risk factors. The ability to process time series data using machine learning techniques to make predictions could greatly improve clinician's reactivity by supplying them with a fast and precise second opinion. Additional applications also exist, such as providing a hospital with insight into their ICU mortality performance compared with other hospitals. Hospitals could achieve this insight by utilising machine learning predictions to compare the likelihood that each patient should've survived versus the real survival outcome. The field of ICU mortality predictions using machine learning has produced extensive evidence of machine learning models outperforming traditional models and having the potential for better interpretability. Beyond the accuracy improvements, the capacity of automation alone provides a strong incentive to use machine learning models for future ICU mortality predictions. This review looks at past methods for ICU mortality risk predictions and compares them with the newest methods.

Traditional risk predictions

Most of the risk tools such as APACHE, Simplified Acute Physiology Score (SAPS) and the Mortality Probability Model (MPM) are scores that have existed and evolved since the 1980s, and have now been updated to adapt to changes in treatment quality since then (Salluh & Soares, 2014). Many of these methods rely on weighted marks, where each medical variable has ranges that are assigned different risk levels. The total amount of marks produce an assessment of risk for the individual, estimating how critically ill the patient is and therefore how critically care is needed. Some of the modern versions of tools use more than 100 variables from the day of ICU admission such as APACHE IV, whilst others use less and more immediate recordings to develop an initial assessment such as SAPS 3 (Salluh & Soares, 2014). Since it takes time to manually input data, the faster and less accurate options are sometimes the only option for medical staff. Despite the manual constraint, traditional techniques have the advantage of interpretability, as looking at where the marks arise from can give the user an indication of what is causing the patient's risk. These scores have proven sufficient for general and specific admission, a study (Silva et al. 2020) found that for acute kidney failure many of the scores were adequate, however improvements in accuracy could still reduce deaths further. Recent research suggests that accuracy can be improved on, as the traditional methods are outclassed in the majority of cases by modern machine learning techniques (Kang et al., 2020; Wang et al., 2022). Kang et al. (2020) looked specifically at patients suffering from kidney injuries and found that both SOFA and APACHE were not able to match the effectiveness of any of the machine learning models they used. Similarly, Wang et al. (2022) found that the best performing predictive models were using XGBoost and LightGBM techniques which far outpaced five traditional methods they tested for general ICU mortality predictions. Although a newer version of the traditional models could temporarily outperform the machine learning techniques, adaptivity in the machine learning techniques means that it could potentially learn from data as it makes predictions. Since machine learning models can be explainable, it may also be more interpretable to have a new model. The potential for a higher quality prediction system means that machine learning techniques could make better predictions for ICU mortality than traditional methods, saving lives by providing better risk information to medical professionals.

PhysioNet Challenge Attempts

Teams across the world competed in the PhysioNet challenge (Silva et al. 2012). The datasets used in the challenge contained up to 37 time series variables and 3 single value variables. Across all the papers, we can see a variety of pre-processing choices. Whilst some teams (Lee et al. 2012) simply selected the last measurement of each of the variables, others such as Hamilton & Hamilton (2012) and Johnson et al. (2012) split each time series variable into minimum, maximum, median, first and last and more, which allowed a model to consider each of those features separately for every time series variable. None of the high scoring challenge attempts directly analysed the time series data, likely due to the fact that full time series analysis is often excessive in the datasets, with less than five readings per variable most of the time. There were fewer machine learning models available to the groups at this time and generally models were less capable. Consequently, they often used limited variations on standard tree (Johnson et al. 2012) and logistic regression models (Lee et al. 2012). The limited and weaker machine learning options imply that even with good pre-processing, the accuracy for these original experiments is likely worse than what modern techniques could achieve. However, many of the top scoring teams still received better results for accuracy than SAPS by direct comparison. Missing data was also often present in the datasets and filling the data or removing it entirely were decisions that had to be made with the training set. One group (Lee et al. 2012) used age and sex to estimate values where values were missing, and discarded data missing age or sex. Whereas another group substituted any missing values with estimates from all other values (Johnson et al. 2012). In order to ensure validity in the training set, both of the previously mentioned approaches may be biassing the data by over-emphasising known trends. Thus, careful consideration regarding any data cleaning for modern algorithms should be taken to avoid overfitting. The pre-processing techniques from the PhysioNet challenge papers are ultimately still applicable today, and can be used with new machine learning techniques to find a highly accurate model. Conversely, the machine learning techniques used were limited and should not be the only methods applied in modern approaches.

Recent Machine Learning models

Many recent models are specific to different conditions such as kidney problems (Kang et al. 2020) and sepsis (Jiang et al. 2021), however there's also general predictive algorithms such as with Wang et al. (2022) and Gonzales-Novoa et al. (2021). One method (Gonzales-Novoa et al. 2021) involves a machine learning algorithm that not only automatically intakes data and classifies risk but also actively alerts medical personnel when the risk reaches a certain level. Both this method and the Jiang et al. (2021) method include explanations as produced by SHapley Additive exPlanations (SHAP) to inform the medical professionals about why the risk is high, which provides much more value for the doctor than a simple risk assessment and is necessary for machine learning methods to outpace the traditional methods. Another study (Pan et al., 2020) focused on COVID-19 ICU assessment used both Local Interpretable Model-Agnostic Explanations (LIME) and SHAP to provide interpretable analysis. From the studies researched, most explainable implementations use one or both of LIME and SHAP to achieve interpretability. Many of the recent methods have specialised purposes for specific illnesses, which tests the exhaustiveness of the new machine learning models and whether they can approach all common and uncommon illnesses in ICU scenarios.

Despite the promising results of current research, the automation of models whilst maintaining interpretability remains a considerable challenge. However, one recent success used an additional layer in a neural network to track the influence of different variables in a cardiovascular focused ICU study (Gandin et al., 2021). The ability to employ an algorithm that can adapt whilst also providing insights in an explainable way would strongly benefit clinicians in the field by reducing the time wasted on diagnosis and manual monitoring, which explains the use of LIME and SHAP in many implementations. The ability to integrate models into pre-existing systems is also paramount to the success of the machine learning implementations, as this is what makes active alerts possible for a patient's condition (Luo et al., 2022). The ability to integrate existing systems with newer machine learning models is somewhat more difficult to generalise than the machine learning model itself, and may require additional effort. Many of the modern approaches discussed so far have used XGBoost, LightGBM, Short Vector Machine, Logistic regression or decision trees as their machine learning models. One implementation (Lan et al., 2022) utilised four different machine learning algorithms to explore the different accuracies, selecting the most effective algorithm. Testing several different approaches and comparing performances with known test data appears greatly beneficial regardless of the specific illness being analysed. Given the range of techniques used, it appears that attempting multiple different techniques in order to find the best model would be optimal. The implementations are relatively interchangeable, only with potential variations in the technique used to make the model explainable.

Conclusion

The traditional techniques used in ICU mortality prediction such as APACHE and SOFA are outclassed by modern machine learning approaches. In order to increase accuracy of prediction rates and reduce death rates, the best possible models must be available to our medical professionals. The PhysioNet challenge prediction model attempts made some good pre-processing choices that may still be usable today. However, more recent methods have been more effective due to advancements in machine learning techniques. Innovation in the ways that we integrate the machine learning models into existing systems may be necessary for future developments to thrive, as the improved performance is clear, however the implementation is not always as straight forward. Regardless, the machine learning techniques in combination with SHAP or LIME can provide a highly interpretable, accurate and fast model which could allow medical professionals to greatly improve their efficiency in assessment and as a consequence, treatment in general. Based on the studies explored in this review, machine learning based risk prediction software is worth exploration and has recently outperformed alternative methods.

Outcomes

What has been implemented

Our group has implemented a web application where ICU staff can create an account, login to their account, and upload their hospital's ICU patient data to receive performance analysis. During the file upload process, the program collects the data, wrangles the dataset, and imputes missing values in the dataset. The program then uses a Binary Neural Network from Sklearn and Tensorflow's Python library previously trained on 12,000 patients uploaded for the PhysioNet ICU to predict the outcomes of the end user's patients. Following the completion of the data wrangling, imputation and analysis in the back end of the program, the user is directed to an analysis page that displays the total patients, the total patients that actually survived, the total patients that were predicted to survive, the overall score of the hospitals performance, and a table that displays each patient's ID, actual outcome, predicted outcome, likelihood of outcome, and a link to LIME analysis for that patient. The end user can select to view LIME analysis for any individual patient to determine what variables have contributed to that prediction. The end user can login to their account to access any previously uploaded analysis and download any previous file uploads via the web applications history page. Once the end user has finished using the web application they can choose to logout.

How are requirements met and the results achieved/product delivered

The project requirements provided in the project specifications were focused on the accuracy, interpretability, and useability of the program. In terms of accuracy, the project aimed to develop a model prototype that can predict mortality of ICU patients in an accurate way. Two ways to measure the performance of our machine learning model were used in the PhysioNet ICU Challenge. The first method used the minimum value out of Precision and Sensitivity for predicting deaths. The second method used a metric based on the Hosmer-Lemeshow H statistic. To calculate the H statistic for a given entry, the in-hospital mortality risks predicted by that entry are first sorted and the corresponding records are binned into deciles designated by *g = 1,2,3...10*. Thus, the first decile (*g = 1*) of the 2990-record set X contains the 299 records with the lowest predicted risk, the second decile contains the next 299 records, etc. The H statistic is then calculated as:

$$H = \sum_{g=1}^{10} \frac{\left(O_g - E_g\right)^2}{N_g \pi_g \left(1 - \pi_g\right) + 0.001}$$

Where for each decile g, O_g is the observed number of in-hospital deaths, E_g is the predicted number of deaths, N_g is the number of records (299), and π_g is the mean estimated risk for records in the decile. For the first method of scoring, a larger value indicates better accuracy. For the second method of scoring, a lower value indicates better accuracy. Commonly used ICU prognostic model SAPS-I scored 0.3125 in the first method of scoring and 68.58 in the second method of scoring. Our predictive model scored 0.59 in the first method of scoring and 37.49 in the second method of scoring. The best performing entry into the PhysioNet ICU Challenge scored 0.5353 in the first method of scoring and 29.86 in the second method of scoring. This demonstrates that our predictive model's accuracy considerably exceeded the performance of industry standard prognostic models. Additionally, our predictive model would have been a top performing contestant in the PhysioNet ICU Challenge. Thus, our project exceeded its accuracy requirements.

In terms of interpretability, the project requirements were to provide local interpretable explanations for these predictions. Our project configured LIME to provide interpretable explanations for each prediction. The Python LIME library is a cutting edge local interpretable model explainer that is incorporated into our web application to provide detailed explanations for why the Binary Neural Network predicted each patient outcome. LIME displays the probability of each outcome (survived or not survived), a breakdown of how values for each predictor variable contributed to the patient surviving or not surviving. The LIME analysis also provides a table displaying the patient's values for every predictor variable. The interpretable analysis provided by LIME allows the end user to identify trends in patient values that suggest the patient's condition is deteriorating. In turn, this will assist the end user in reforming current ICU procedures and demonstrates how the project met its interpretability requirements.

In terms of useability, the project requirements were to implement a stand-alone software that is accessible for end users with no data science experience. The software that this project has delivered is accessible on any web browser and is intuitive and easy to use to maximise the number of potential end users. The software does not require any data wrangling or analysis knowledge as it completes all the data wrangling, imputation, and analysis during the file upload. The combination of providing a web application accessible from any device and not requiring any data wrangling or analysis techniques to operate the program illustrate that the useability project requirement was achieved by this project.

Justification of decisions made

Our group chose to use Flask as the web framework for our web application because it has a shallow learning curve and provided the register, login, file upload, and database requirements that our web application required. Additionally, it allowed our group to essentially run the web application from one file.

Our group decided to use a web application with an account system for each end user so that users can have private access to their previous file uploads and analysis. Additionally, it allows the administrator to monitor who is using the web application, our group believes it provides a better end user experience by encouraging end users to return to the web application. Having a web application with an account system also provides an opportunity to make the web application commercially viable in the future by offering paid subscriptions to access the web applications features.

Our group decided that the data uploaded to the web application would have to be in the same format as the data in the PhysioNet ICU Challenge. This decision was chosen for two reasons. Firstly, the PhysioNet ICU Challenge patient data was taken directly from hospitals and our group has assumed that this format is the easiest format for hospitals to upload their data in. Secondly, this format was chosen as we did not want end users to have to perform their own data wrangling and imputation techniques before uploading their data to the web application. Our group believed that our web application should be a complete data science service and not require any computing skills.

Our group chose to not include automated account deletion and resetting password options. This means that end users must email the admin to action either of these requests. These decisions were made because our group believed that these features were outside of our project requirements within the given time frame.

Our group chose to prioritise accuracy over interpretability when selecting a machine learning algorithm. Alternatively, we could have used a form of logistic regression with high interpretability, which we configured during machine learning model selection; however, the overall accuracy was significantly worse. Thus, the decision to use a Binary Neural Network was made to provide hospitals with as accurate feedback about their performance as possible while maintaining a reasonable level of interpretability for procedural reform.

Discussion of all results

Overall, the web application created for our project emphatically addresses the key project requirements of accuracy, interpretability, and useability. The final product provides end users the opportunity to easily receive ICU performance analysis that is considerably more accurate and advanced than industry standard indicators (such as SAPS-I). Additionally, allowing any user to create an account and access their previous uploads privately increases the pool of potential end users. Thus, allowing users with no data science experience to have access to an advanced machine learning algorithm. This software undoubtedly provides hospitals with information that can significantly improve their reporting and patient care procedures. However, our group understands that this web application is currently only a foundation for a commercially successful web application and requires the addition of multiple software features (discussed in the following sections), improved privacy features (such as obtaining an SSL certificate), more visually palatable analysis, and a faster data processing and imputation process.

Limitations of project outcomes

The file upload and ensuing data wrangling, imputation, and analysis process is slower than it could have been if the initial dataset was already wrangled and imputed. Additionally, the LIME analysis speed is slow compared to opting to choose a highly interpretable machine learning algorithm and displaying similar results. Furthermore, during the process of waiting for the data to upload and the LIME analysis to run, if the end user tries to cancel either process it creates an error.

Overall, the software features are only intended to meet the key project requirements and could be expanded to create a better overall end user experience. This would include adding the ability to verify user accounts using emails, enabling users to reset forgotten passwords via email, and creating an option for users to delete their accounts.

The overall interpretability of results and the visual analysis component of the web application could be improved by creating an analysis page that is more visually appealing.

The web application is currently not commercially viable and would require the addition of multiple features that are addressed in the next section.

Discussion of possible improvements and future works

As aforementioned, it would be useful to provide an account verification feature via email and a password reset option feature via email. Additionally, adding an option for end users to delete their accounts would enable users to create and manage their own account without the need to contact an administrator.

The table that is shown on the analysis page that details the ID, predicted outcome, actual outcome, likelihood of actual outcome, and a link to that patient's LIME explanation could become downloadable. This would enable hospitals to easily conduct reporting based on the findings of the web applications analysis.

The web application could provide more in-depth visual analysis that demonstrates how a hospital's ICU is performing compared to industry standard. This would include a bell curve that takes in data from hospitals around the world and places the current hospital in a particular percentile. Furthermore, the analysis page provided using LIME could be reconfigured to be more visually palatable and in line with the formatting of other web pages.

The web application could create organisation accounts that link all the user accounts associated with that organisation and provide end users varying levels of access depending on their position. This would also allow for the website to add visual customisation by adding in company logos and company colouring for the performance analysis component. This can then be easily used for that company's reporting processes.

If the previous suggestions were added to the web application along with security improvements, it would make the software commercially viable through a subscription model offered to organisations.

Methodology

Design

In our original user flow diagram (appendix 1.1), we specified the approximate outline of our web application. The final version follows a similar structure but includes several additions and changes. Notably the website now includes a signup page, excludes the help page and has a 'history' page unique to each user which stores their uploaded data and provides easy analysis of past uploads.

The Help page was removed from the final web application. We instead provide the user guide for users to learn how to interact with the software. This was done mainly to ensure that the web application itself was straightforward and to maximise our effort on the practical components of the application.

The History page was added during web application production, as it became clear that file management for several datasets could begin to get difficult to manage for the user, so we opted to provide that service from the web application, also providing an easy way for the developers to access any uploaded data if needed.

The analysis output was replaced with an analysis page and analysis_id page. The analysis page is a full page dedicated to a report for the hospital on their performance and the analysis_id page is specific patient analysis, looking at the contributions of each factor to the predicted outcome for the patient. This change was made to simplify the overall structure of the application and has little impact on the functionality as proposed in appendix 1.1.

To summarise, the user signs up to the application, logs in, and then uploads a zip file with data in the correct format. They are then shown the analysis for that data, including hyperlinks to specific patient analysis. When the upload of the zip file successfully occurs, the zip is saved in a unique directory to the user, and they can redownload it/reanalyse it from the History tab in the web application. On the software side, the application calls a prebuilt neural network and uses imputation to wrangle the uploaded time series data into the correct format so that the neural network can process it.

Implementation

The web application was produced from Python, primarily using Flask for the application functionality. The machine learning model is produced with sklearn and tensorflow, with LIME providing specific patient analysis (presenting the impact of different factors). Originally Django was going to be used rather than Flask, however it appeared to be easier to coordinate our work with Flask, as we could easily work from essentially a single python file. HTML and CSS however were also used for visual components of the web application, but ultimately most of the web application functionality was produced from one python file, with some functionality included in other secondary python files.

To coordinate our group efforts, Trello and GitHub were used for management of tasks and the project files respectively. Some additional communication tools such as Facebook Messenger, Zoom and Google Drive were used throughout the project.

Software Deliverables

The final product of our project is a web application running using Python and Flask framework. This web application provides the functionality for the intended users for example ICU units to examine how well they performed as compared to the industry standard. The software includes using machine learning packages and algorithms to predict the patients' survival as well as using LIME to make the blackbox algorithm interpretable. Python libraries including Flask, Sklearn, TensorFlow and LIME are used for the software. HTML and CSS are used for building and styling the website.

The web application includes a number of pages with different functionalities. Each page will be explained using screenshots below.

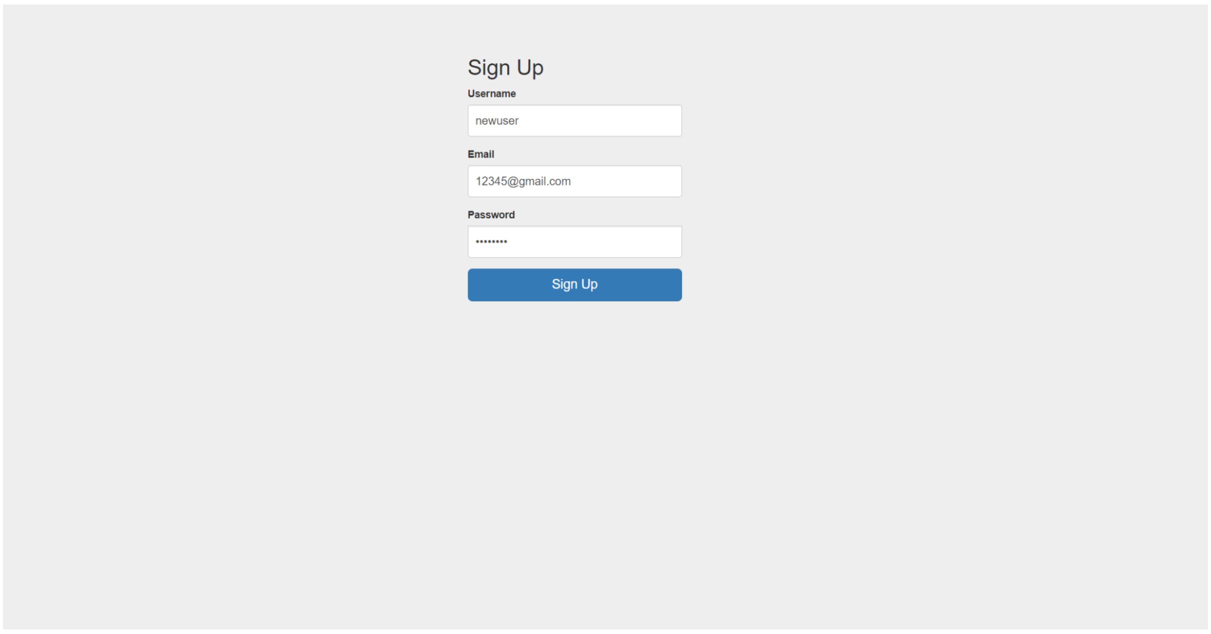


Fig 1

Fig 1 is a screenshot of the sign up page, before any user can use the software, they have to sign up using their email and set a username and password for the account. This account information will then be added into a database. After signup, the users can access their files. Each user has their own storage places and hence the files are not shared among all users. Appendix 5.1 is a screenshot of the source code for the sign up page. It clearly shows how the error is handled when the user input is invald and how the individual storage space is created.

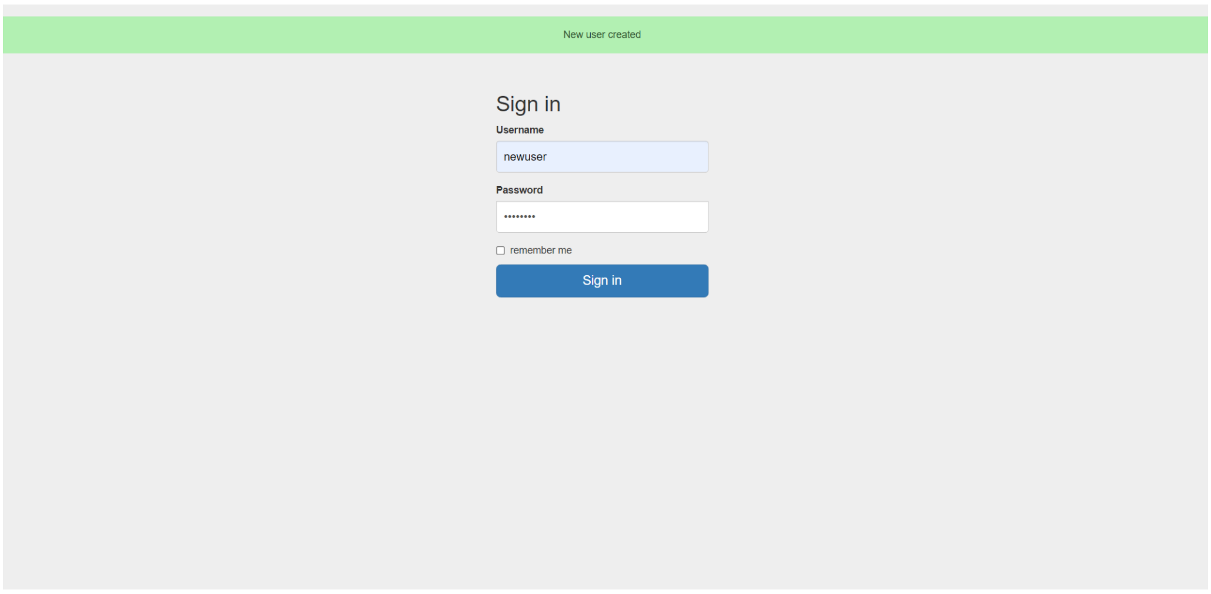


Fig 2

After the sign up is completed, the webpage will be automatically directed to the Sign in page as shown in Fig 2. The users can use the account they just created in the Sign up page to login to the web application. Entering username and password unmatch with the account information in the database will show an error message and the login will not be successful.

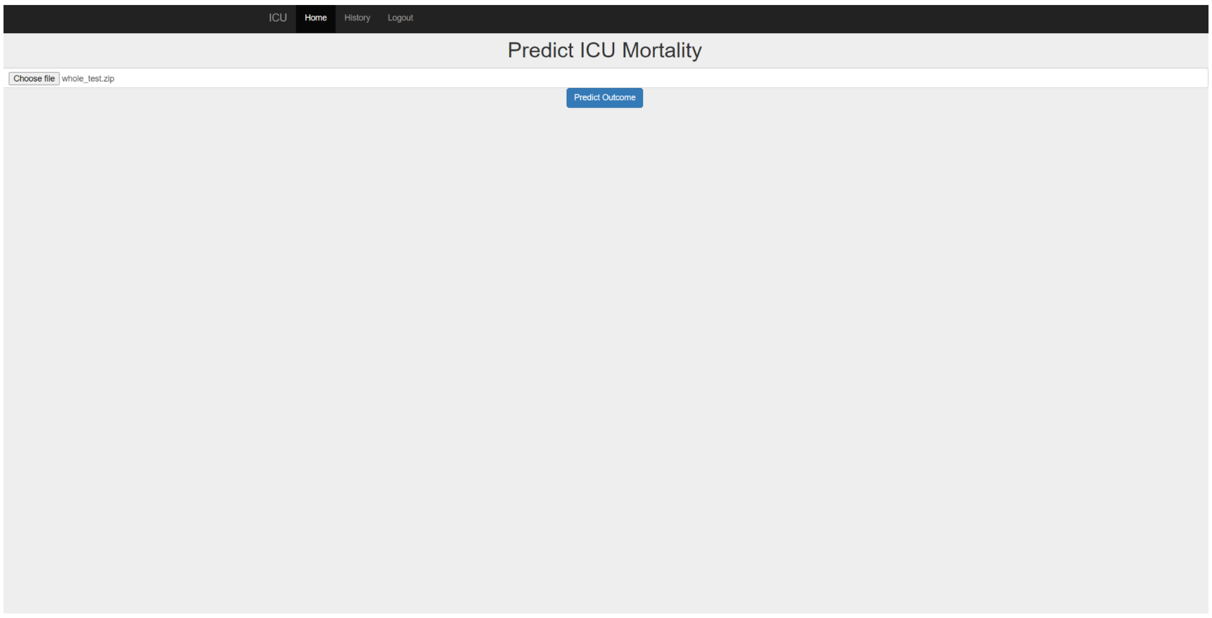
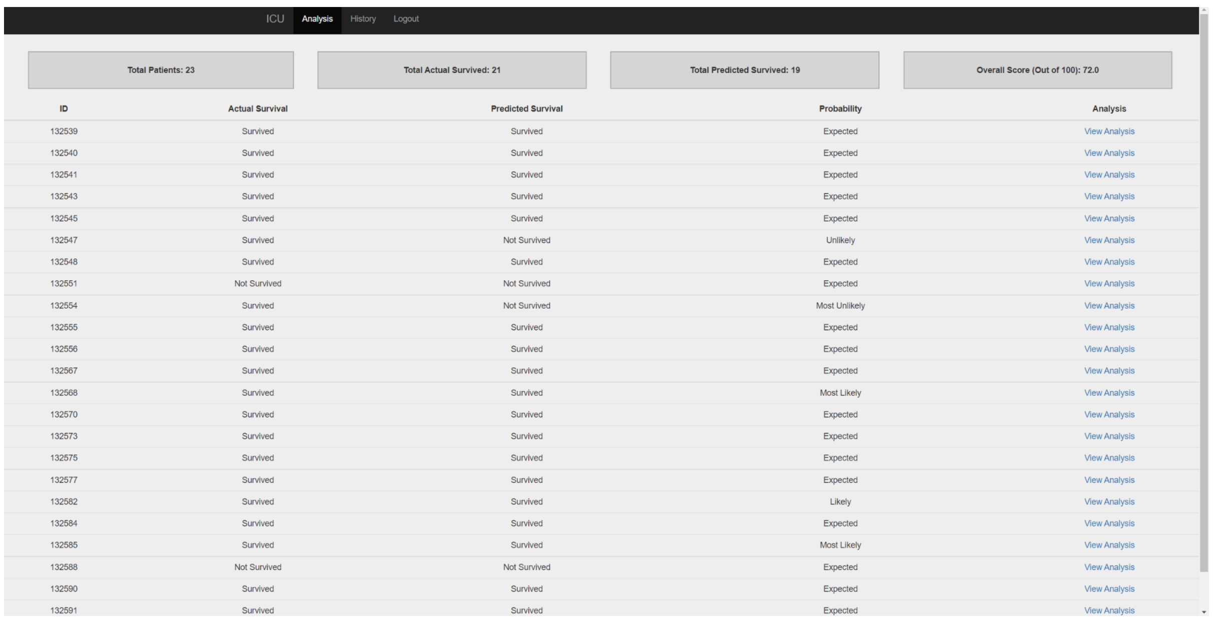


Fig 3

Fig 3 is a File Upload page. You will be directed to here after the log in. The user can upload the data of patients here. The program only takes in a zip file which contains the data of patients in .txt. The format of the patient file is strictly restricted to the same format of the files provided by the PhysioNet Challenge. Uploading a file other than the zip format will not be successful. Click on the "Predict Outcome" button will let the program use a neural network machine learning algorithm to develop a model and then predict the mortality of each patient and take the user to the Analysis page. Appendix 5.2 is a screenshot of the source code for using our neural network model to predict the survival of each patient.



Total Patients: 23

Total Actual Survived: 21

Total Predicted Survived: 19

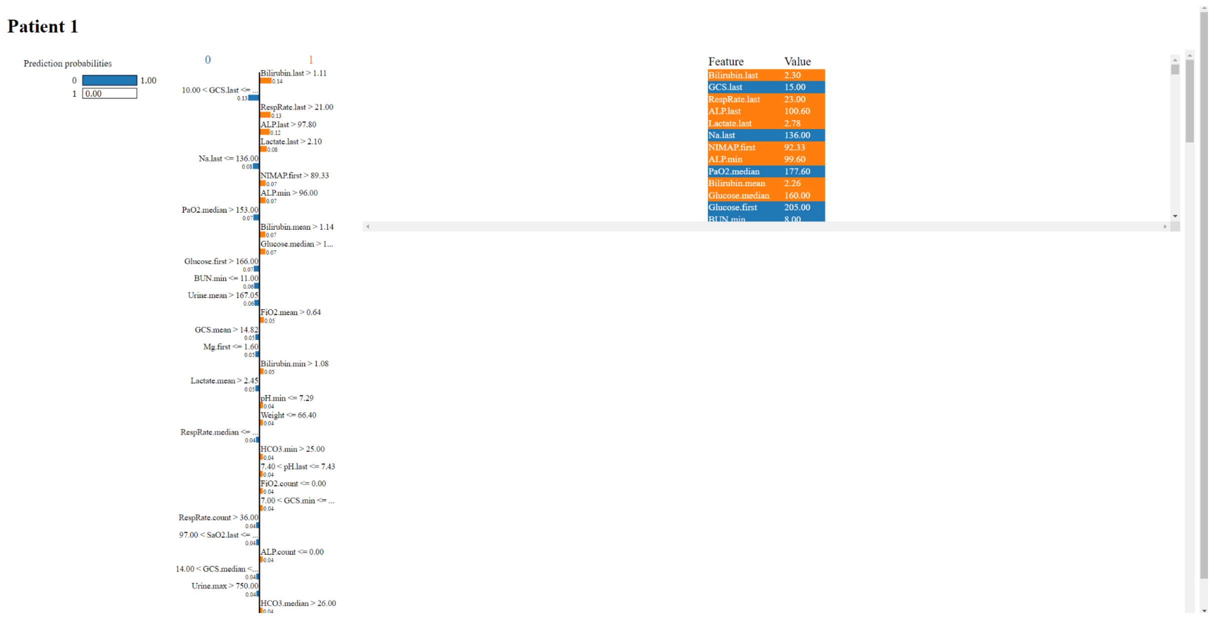
Overall Score (Out of 100): 72.0

ID	Actual Survival	Predicted Survival	Probability	Analysis
132539	Survived	Survived	Expected	View Analysis
132540	Survived	Survived	Expected	View Analysis
132541	Survived	Survived	Expected	View Analysis
132543	Survived	Survived	Expected	View Analysis
132545	Survived	Survived	Expected	View Analysis
132547	Survived	Not Survived	Unlikely	View Analysis
132548	Survived	Survived	Expected	View Analysis
132551	Not Survived	Not Survived	Expected	View Analysis
132554	Survived	Not Survived	Most Unlikely	View Analysis
132555	Survived	Survived	Expected	View Analysis
132556	Survived	Survived	Expected	View Analysis
132567	Survived	Survived	Expected	View Analysis
132568	Survived	Survived	Most Likely	View Analysis
132570	Survived	Survived	Expected	View Analysis
132573	Survived	Survived	Expected	View Analysis
132575	Survived	Survived	Expected	View Analysis
132577	Survived	Survived	Expected	View Analysis
132582	Survived	Survived	Likely	View Analysis
132584	Survived	Survived	Expected	View Analysis
132585	Survived	Survived	Most Likely	View Analysis
132588	Not Survived	Not Survived	Expected	View Analysis
132590	Survived	Survived	Expected	View Analysis
132591	Survived	Survived	Expected	View Analysis

Fig 4

Fig 4 is a screen shot of the Analysis page with test data uploaded. The Analysis page lets the user analyse how well their ICU units are performing as compared to the industry standards. We have a few columns on this page. The first column is the patient id which is directly taken from the file name of each patient in the zip file. The second column is whether the patient actually survives in the ICU, also provided by the file uploaded before. The third column is what our machine learning model predicts the mortality of the patients. Next column uses the probability of mortality which the model provides to divide the predicted outcome into a few categories. "Expected" for probability > 95%, "Most Likely" for probability > 80%, "Likely" for probability > 50%, "Unlikely" for probability > 20%, "Most Unlikely" for probability > 5% and "Unexpected" for probability > 0%.

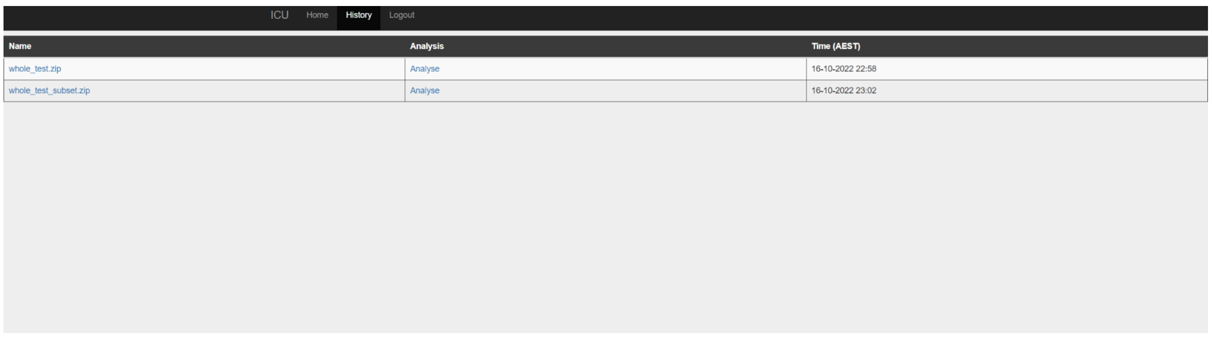
On the top right hand corner of the page, there is also an Overall Score calculated out of 100. This score indicates how well the ICU is performing overall. If the patient's actual survival matches the predicted survival, the score will go up. Furthermore , for instance if the model predicts the patient to be unlikely to survive but actually survived, the score will be higher to show that the ICU is performing well. Appendix 5.3 is a screenshot for computing the score, the full computation involves many lines of code and only the final two lines of codes are shown here.



Feature	Value
Bilirubin.last	2.30
GCS.last	15.00
RespRate.last	23.00
ALP.last	100.60
Lactate.last	2.78
Na.last	136.00
NIMAP.first	92.33
ALP.min	99.60
PaO2.median	177.60
Bilirubin.mean	2.26
Glucose.median	160.00
Glucose.first	205.00
BUN.min	8.00

Fig 5

Click on the View Analysis button will show the LIME interpreted result on a new tab as in Fig 5. Here the user can view more detailed analysis such as which variables affect the mortality the most. Appendix 5.4 is a screenshot of the source code for saving the explanation of LIME. The function is then called by the code in Appendix 5.5. Appendix 5.5 shows how the program produces the LIME interpretation using the datasets and renders the LIME page.



Name	Analysis	Time (AEST)
whole_test.zip	Analyse	16-10-2022 22:58
whole_test_subset.zip	Analyse	16-10-2022 23:02

Fig 6

At last Fig 6 shows a history page where all the uploaded files for the user are listed here. However, all the files with unsuccessful uploads will not be displayed here. Click on the file name will redownload the file to the local storage. Click on the "Analyse" button will direct the user to the Analysis page according to the file selected, and displaying the outcomes.

While implementing the web application, we aim to maintain high software qualities. We chose to use Flask as the framework for our web application as it is a mature and easy to use framework for web development. This largely increased the robustness of our software and saved us a lot of effort.

The software also considers the security of user information as one of the priorities. Since the data of patients can be confidential and important, the user may not want that information to be seen publicly. The software creates separate storage spaces for every user account. One account can only access the files uploaded to the same account and there is no way to access the uploaded files of other users. If different people from the same hospital wish to share the files, using the same account for the entire hospital is recommended.

To increase the usability, we decided to make a web application which can be easily accessed using any web browser. No additional software or environment installation is required. This not only makes the software more accessible, but also can maximise the number of potential end users. Using the software is also considerably easy as it only requires the user to sign in and upload. Everything else is completed automatically in the back end. Furthermore, The datasets uploaded do not need to be wrangled and can contain missing values. This process is completed using basic data wrangling techniques and KNN imputation.

Since we are using Flask as a framework for the web application, scalability will not be an issue. Flask is a microframework, which means that our web application can grow very quickly. Even though our software is small for now, if we want to grow the size of the software, Flask enables us to achieve this easily. The program will run smoothly even if it scales up quickly.

Furthermore, Flask is a template engine. A template engine makes the user able to set a basic layout for all pages of a website and take less effort when making changes to the website. Hence, it saves us a lot of time when we want to update the web application or if we want to maintain it.

However, the shortcoming of our software is that, the running speed for wrangling the data and computing missing values, as well as producing the LIME result depends largely on the machine which the codes are run on. The time can be very long if a computer with bad hardwares is used to run the software. Making a server to run the codes can be a solution to this problem.

Critical Discussion

The project is successful and executed well overall. The whole project was carried out with no major deviations from the initial proposal. There are slight changes on the planning, features and software used but those changes are made due to additional improvements.

As compared to the initial proposal, the time allocation for tasks has changed moderately. This is due to the lack of consideration of additional in-semester assessments such as presentations and reports in semester 2 aside from the actual project itself. This made us have to come up with a new plan with more details at the start of semester 2. We used Trello to track the tasks due date using the new planning. Furthermore, we chose to use a Waterfall project management methodology which means that we cannot start the next step without the completion of the previous steps. Due to the lack of experience in this project management methodology and the unexpected business at the start of the semester, we were slightly behind schedule at the beginning. Besides these, for data wrangling and implementing machine learning algorithms, every team member was asked to try these simultaneously. After all have completed, we then compare the results as a group and choose a best performing method or algorithm. These tasks also deviate a bit from the original planning as some cannot finish the task by the predefined date due to other circumstances.This resulted in us having to defer all the following tasks. What we did well to cope with this problem is, we anticipated that we might encounter this kind of problem so we left some buffer times while we were making the plan. This made it not necessary for us to rush through the tasks once we were behind the schedule.

Another part we deviated from the original design is the software framework used. At the beginning of semester 1, we considered using RShiny but then quickly replaced it with Django. However, we as a group did not use any of these web development frameworks before and we did not know what is the best choice for our project. Every group member spent a lot of time and effort to learn Django from scratch, but then realised that Django may not be the most suitable framework to use. We then found a more suitable and easier to use framework, which is Flask, for our web application. We had to learn Flask again before we could implement the actual software. This caused a lot of waste in time. This situation could be avoided if we have more experience in web development.

There are some software or packages we did not use in the final product. For instance, we decided to use R to build our predictive model in our initial proposal. However, we found out that the packages provided in Python are powerful and efficient enough for our model. With addition to the fact that we are using Python and Flask to build our web application already, using R seems redundant. We also decided to use packages like ggplot2 to visualise the result. But we quickly realised that the blackbox algorithm cannot be easily visualised using the traditional plotting packages and we then use entirely LIME to interpret the result.

When we were planning the project in semester 1, we wanted to add in a lot of features but now we realised that adding those features which are not the main focus of the software will only decrease the usability. We wish our program is easy to use for our intended user and thus we should pay more attention on how to improve the main functionality instead of other features.

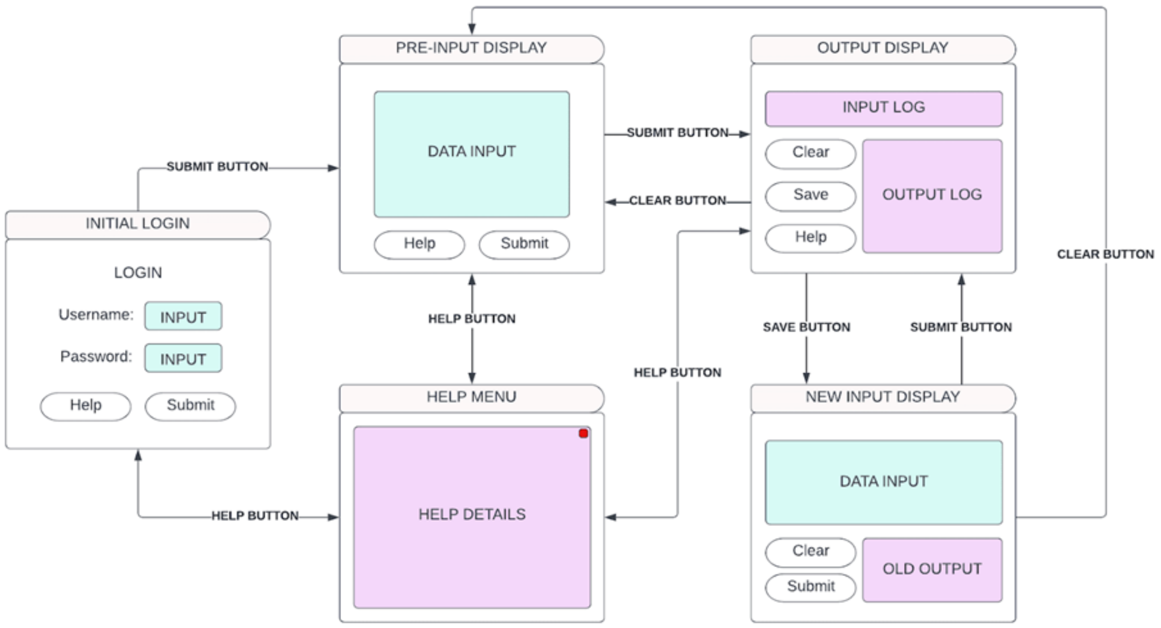
In conclusion, the project went very well. We encountered some challenges but were able to overcome them quickly. The original intended outcome and deliverables were achieved well with some improvements. There are still some limitations of the software which we cannot avoid, but we did our best to minimise them. For example, for the long running speed for wrangling the data, we could have restricted the input files even more so that we did not need so much time to compute missing values. But doing so will decrease the usability and hence we chose to sacrifice the running time.

Conclusion

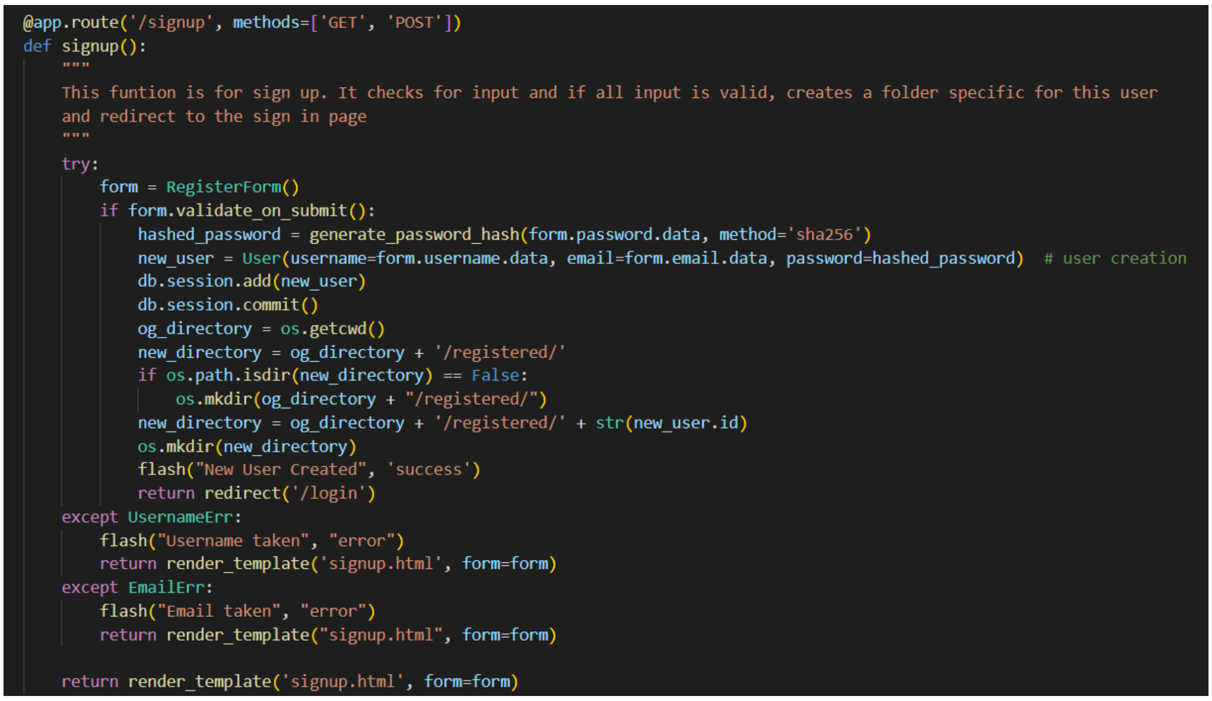
It is evident that there is a need for a more accurate and interpretable predictive model for mortality in ICU. The web application created for our project emphatically addresses the key project requirements of accuracy, interpretability, and useability. The project's accuracy requirement was achieved by the significant improvement in accuracy compared to industry standard indicators (such as SAPS-I) and the entries to the PhysioNet ICU Challenge. Furthermore, the project's interpretability requirement was met through the addition of LIME, which provides a level of interpretability that can benefit hospitals in creating procedural and reporting reform. Finally, the project's useability requirement was met by making the web application available on any web browser for individuals with novice level computing skills. Using Python, Flask, Tensorflow, SKlearn, HTML, CSS, Pandas, and SQLAlchemy, the web application is easily administered with beginner level software development experience. Ultimately, the project achieved its desired outcomes to deliver a software program that can assist hospitals in improving ICU patient care. However, this web application is currently only the foundation of a commercially successful web application. To make this web application commercially viable, it requires the addition of multiple software features, improved privacy features, more visually palatable analysis, and a faster data processing and imputation process.

Appendix

1.1 Initial User Flow Diagram

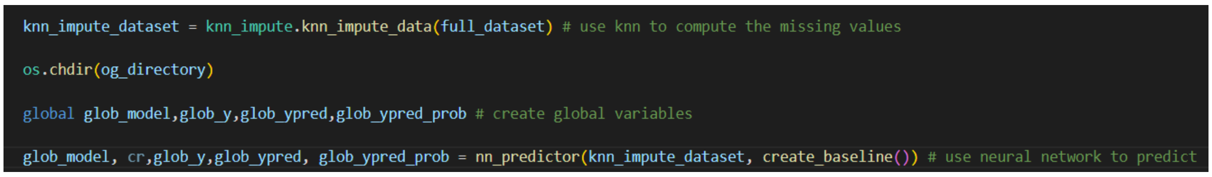


5.1 Source code for Sign up page



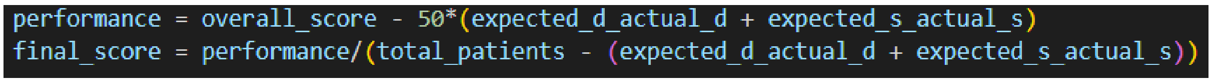
```
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    """
    This funtion is for sign up. It checks for input and if all input is valid, creates a folder specific for this user
    and redirect to the sign in page
    """
    try:
        form = RegisterForm()
        if form.validate_on_submit():
            hashed_password = generate_password_hash(form.password.data, method='sha256')
            new_user = User(username=form.username.data, email=form.email.data, password=hashed_password)  # user creation
            db.session.add(new_user)
            db.session.commit()
            og_directory = os.getcwd()
            new_directory = og_directory + '/registered/'
            if os.path.isdir(new_directory) == False:
                os.mkdir(og_directory + "/registered/")
            new_directory = og_directory + '/registered/' + str(new_user.id)
            os.mkdir(new_directory)
            flash("New User Created", 'success')
            return redirect('/login')
    except UsernameErr:
        flash("Username taken", "error")
        return render_template('signup.html', form=form)
    except EmailErr:
        flash("Email taken", "error")
        return render_template("signup.html", form=form)

    return render_template('signup.html', form=form)
```

5.2 Source code for predicting mortality



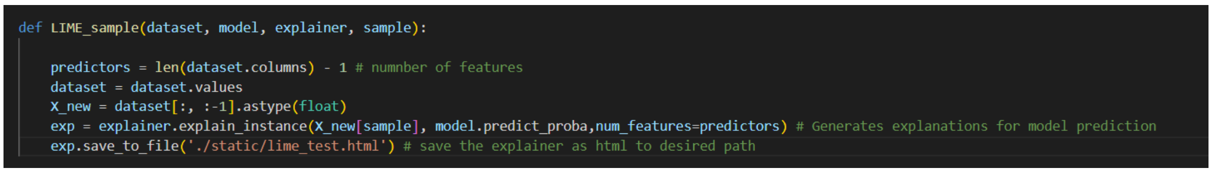
```python
knn_impute_dataset = knn_impute.knn_impute_data(full_dataset) # use knn to compute the missing values

os.chdir(og_directory)

global glob_model,glob_y,glob_ypred,glob_ypred_prob # create global variables

glob_model, cr,glob_y,glob_ypred, glob_ypred_prob = nn_predictor(knn_impute_dataset, create_baseline()) # use neural network to predict
```

5.3 Source code for computing the score in Analysis page



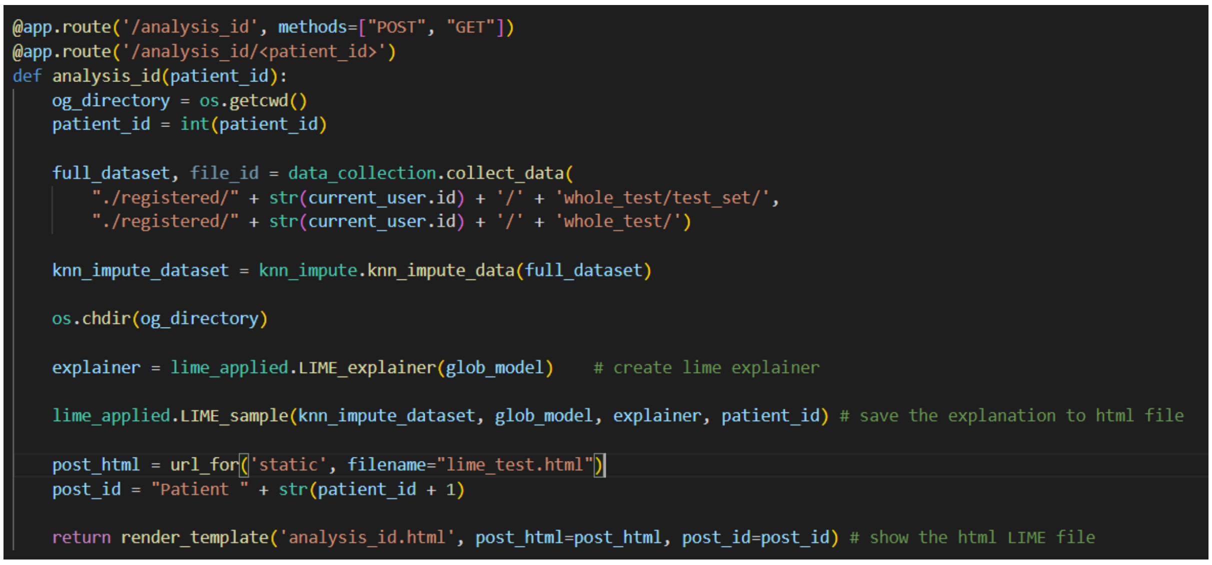
```python
performance = overall_score - 50*(expected_d_actual_d + expected_s_actual_s)
final_score = performance/(total_patients - (expected_d_actual_d + expected_s_actual_s))
```

5.4 Source code for saving the explanation for LIME explainer



```python
def LIME_sample(dataset, model, explainer, sample):

    predictors = len(dataset.columns) - 1 # numnber of features
    dataset = dataset.values
    X_new = dataset[:, :-1].astype(float)
    exp = explainer.explain_instance(X_new[sample], model.predict_proba,num_features=predictors) # Generates explanations for model prediction
    exp.save_to_file('./static/lime_test.html') # save the explainer as html to desired path
```

5.5 Source code for building the LIME page



```python
@app.route('/analysis_id', methods=["POST", "GET"])
@app.route('/analysis_id/<patient_id>')
def analysis_id(patient_id):
    og_directory = os.getcwd()
    patient_id = int(patient_id)

    full_dataset, file_id = data_collection.collect_data(
        "./registered/" + str(current_user.id) + '/' + 'whole_test/test_set/',
        "./registered/" + str(current_user.id) + '/' + 'whole_test/')

    knn_impute_dataset = knn_impute.knn_impute_data(full_dataset)

    os.chdir(og_directory)

    explainer = lime_applied.LIME_explainer(glob_model)    # create lime explainer

    lime_applied.LIME_sample(knn_impute_dataset, glob_model, explainer, patient_id) # save the explanation to html file

    post_html = url_for('static', filename="lime_test.html")
    post_id = "Patient " + str(patient_id + 1)

    return render_template('analysis_id.html', post_html=post_html, post_id=post_id) # show the html LIME file
```

References

Bannerman, P. (2008). Risk and risk management in software projects: A reassessment. *Journal Of Systems And Software, 81*(12), 2118-2133. doi: 10.1016/j.jss.2008.03.059

Boehm, B. (1991). Software risk management: principles and practices. *IEEE Software, 8*(1), 32-41. doi: 10.1109/52.62930

Costa e Silva, V., de Castro, I., Liano, F., Muriel, A., Rodriguez-Palomares, J., & Yu, L. (2011). Performance of the third-generation models of severity scoring systems (APACHE IV, SAPS 3 and MPM-III) in acute kidney injury critically ill patients. *Nephrology Dialysis Transplantation, 26*(12), 3894-3901. https://doi.org/10.1093/ndt/gfr201

Gandin, I., Scagnetto, A., Romani, S., & Barbati, G. (2021). Interpretability of time-series deep learning models: A study in cardiovascular patients admitted to Intensive care unit. *Journal Of Biomedical Informatics*, *121*, 103876. https://doi.org/10.1016/j.jbi.2021.103876

González-Nóvoa, J., Busto, L., Rodríguez-Andina, J., Fariña, J., Segura, M., & Gómez, V. et al. (2021). Using Explainable Machine Learning to Improve Intensive Care Unit Alarm Systems. *Sensors, 21*(21), 7125. https://doi.org/10.3390/s21217125

Jiang, Z., Bo, L., Xu, Z., Song, Y., Wang, J., & Wen, P. et al. (2021). An explainable machine learning algorithm for risk factor analysis of in-hospital mortality in sepsis survivors with ICU readmission. *Computer Methods And Programs In Biomedicine, 204*, 106040. https://doi.org/10.1016/j.cmpb.2021.106040

Johnson, A., Dunkley, N., Mayaud, L., Tsanas, A., Kramer, A., & Clifford, G. (2012). Patient Specific Predictions in the Intensive Care Unit Using a Bayesian Ensemble. *2012 Computing In Cardiology, 39*, 249-252. Retrieved 18 May 2022, from https://ieeexplore.ieee.org/document/6420377.

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, 24*(1), 1-9. https://doi.org/10.1186/s13054-020-2752-7

Kwak, Y., & Stoddard, J. (2004). Project risk management: lessons learned from software development environment. *Technovation, 24*(11), 915-920. doi: 10.1016/s0166-4972(03)00033-6

Lan, L., Chen, F., Luo, J., Li, M., Hao, X., Hu, Y., Yin, J., Zhu, T., &amp; Zhou, X. (2022). Prediction of intensive care unit admission (&gt;24h) after surgery in elective noncardiac surgical patients using machine learning algorithms. DIGITAL HEALTH, 8, 205520762211105. https://doi.org/10.1177/20552076221110543

Lee, C., Arzeno, N., Ho, J., Vikalo, H., & Ghosh, J. (2012). An Imputation-Enhanced Algorithm for ICU Mortality Prediction 2012 Computing In Cardiology, 39, 253-256. Retrieved 18 May 2022, from https://ieeexplore.ieee.org/document/6420378.

Luo, C., Zhu, Y., Zhu, Z., Li, R., Chen, G., &amp; Wang, Z. (2022). A machine learning-based risk stratification tool for in-hospital mortality of intensive care unit patients with heart failure. Journal of Translational Medicine, 20(1). https://doi.org/10.1186/s12967-022-03340-8

McBride, T. (2008). The mechanisms of project management of software development. *Journal Of Systems And Software, 81*(12), 2386-2395. doi: 10.1016/j.jss.2008.06.015

Pan, P., Li, Y., Xiao, Y., Han, B., Su, L., & Su, M. et al. (2020). Prognostic Assessment of COVID-19 in the Intensive Care Unit by Machine Learning Methods: Model Development and Validation. *Journal Of Medical Internet Research, 22*(11), e23128. https://doi.org/10.2196/23128

Salluh, J., & Soares, M. (2014). ICU severity of illness scores. *Current Opinion In Critical Care, 20*(5), 557-565. https://doi.org/10.1097/mcc.0000000000000135

Silva, I., Moody, G., Scott, D. J., Celi, L. A., & Mark, R. G. (2012). Predicting In-Hospital Mortality of ICU Patients: The PhysioNet/Computing in Cardiology Challenge 2012. *Computing in cardiology, 39*, 245–248.

Tavares, B., da Silva, C., & de Souza, A. (2017). Risk management analysis in Scrum software projects. *International Transactions In Operational Research*, *26*(5), 1884-1905. doi: 10.1111/itor.12401

Wang, P., Xu, J., Wang, C., Zhang, G., & Wang, H. (2022). Method of non-invasive parameters for predicting the probability of early in-hospital death of patients in intensive care unit. *Biomedical Signal Processing And Control, 73*, 103405.

https://doi.org/10.1016/j.bspc.2021.103405

Team Member Annex

Team Member Name	**Sections Contributed**	**Overall Contribution %**
Daniel Clark	Background (10) Methodology (15) References (5)	**33.3%**
Li Yunhao	Critical Discussion (10) Software Deliverables (15) Code Listing in Appendix (5)	**33.3%**
Harry Lawson	Front Cover Sheet (2) Introduction (8) Outcomes (20) Conclusion (5) Style and Presentation (5)	**33.3%**