Mortality and Length-of-stay (LOS) prediction

December 2022

Main content: 10 pages; Reference: 1 page

STAT3612 Statistical Learning
Department of Statistics Actuarial Science Faculty of Science
the University of Hong Kong

FAN Zixian
Kuang Yuzhi
Hu Yuxin
Li Xiong
Zhao Yiming

Course Co-ordinator Dr. Yu

1 Introduction

The project aims to obtain a training model for Mortality and Length-of-Stay (LOS) and make predictions based on data from an intensive care unit in Boston. Data processing is applied from the perspective of reducing meaningless data, improving model training data, and predicting accuracy. With the obtained data sets, Mortality prediction (task 1) and LOS prediction (task 2) are conducted. In task 1, the model's prediction is a binary prediction of mortality, which is significant for patients and hospitals. For example, from the expense perspective, a study showed that this variable had a substantial relationship with ICU inpatient costs (Thompson et al., 2015). As for task 2, we focus on the regression problem of predicting patients' length-of-stay using the same X data. LOS is defined as "the length of an inpatient episode of care, calculated from the day of admission to the day of discharge, and based on the number of nights spent in hospital" (Segen's Medical Dictionary, 2011). It is a significant indicator of management efficiency, quality of patient care, and operation assessment in hospitals. By accurately predicting the LOS of patients, hospitals could manage accordingly to reduce their costs and improve patient satisfaction (Chrusciel et al., 2021). Given the importance of Mortality and LOS prediction, we train different machine learning models to predict and assess their performance using ROC_AUC and root mean square error (RMSE) as the scores, respectively. In addition, we also investigate the reason why a model performs well or poorly, and discuss some important factors.

2 Data analysis and process

2.1 Data analysis

The data used in the study came from the collection of over 50,000 patients in an intensive care unit, split into three parts after preprocessing: training, validation, and test data. Furthermore, the dependent variable can be divided into two parts according to Mortality and Length-of-stay (LOS). Among the independent variables, the shape of the training data set is (16760, 7488), which can be divided into "Mean," "Mask," and "Time_Since_Measured " three parts, each with 2496 columns. It is worth mentioning that Mask = 0 means the data has a missing mean, so there are two approaches to fill in these values. The first one is to determine it according to the forward-filled method considering other values of Mask = 1, while another technique considers to fulfill it by the global set means when the first method doesn't work. Moreover, those three parts continue to be split into 24 hours daily. Therefore, when we consider only the independent variable types, there are only 104 columns in total. The 24-hour data corresponding to each independent variable can be treated in mean and extreme two approaches. The mean is an intuitive treatment because it shows whether the average of that variable is higher than the population norm for that period. The extreme variation is a special understanding that a patient's vital status may change dramatically due to a sudden increase or decrease in a variable at a given time, as in other studies where there may be rapid, and large fluctuations in hemodynamics during an outbreak of pulmonary artery disease (Caner & Hansen, 2004), and such sudden changes are not shown in the mean values. The raw number of the validation set and the test set are 2394 and 4790, respectively. Regarding the explanatory variables, the Mortality data are binary, while the LOS data are quantitative. After reviewing the structure of values in the train set of Mortality data, it is found that the number of "0" values is 15535, and the number of "1" values is 1225, where there is a critical imbalance problem. Nevertheless, the "0" and "1" values in the validation set are 3106 and 1553, and the two-fold relationship between them is much smaller than the 15-fold in the training set. This difference should generate an inaccuracy classification result when using the fitted model to evaluate the validation set if it is constructed under the train set. In addition, in the aftermath of generating the distribution of the LOS data, it can be diagnosed that there are more values stacked near 0 and fewer values away from 0, which is consistent with the intuition that the patient's stay in the ICU is primarily concentrated in shorter intervals, as most of the patients will either leave with successful resuscitation or unfortunately pass away.

2.2 Data processing

The data processing is firstly done based on the hypothesis, where the data processing steps are divided into three parts: data pre-processing, class imbalance, and data scaling, after which the obtained data will be used in the training and validation sets for trial. The better-performing data sets will be considered for further use, while the poorly performing ones will be improved or discarded after the reasons are identified. The following figures show the study procedure (the parts of the design data processing marked in red) and the data processed step by step:

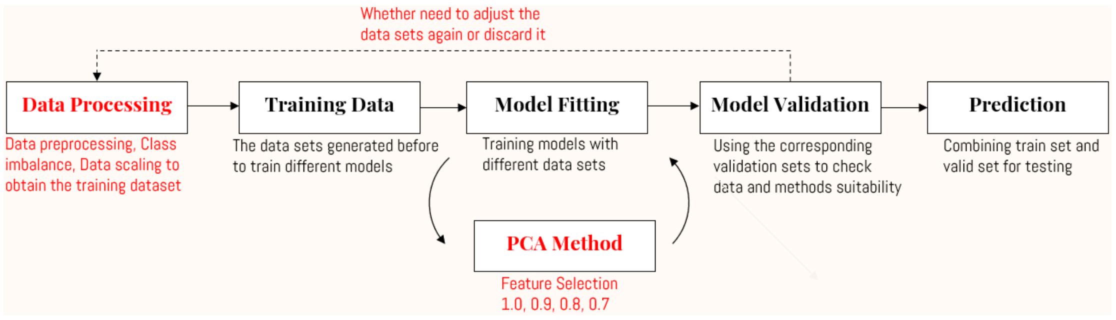
Whether need to adjust the data sets again or discard it
Data Processing
Data preprocessing, Class imbalance, Data scaling to obtain the training dataset
Training Data
The data sets generated before to train different models
Model Fitting
Training models with different data sets
Model Validation
Using the corresponding validation sets to check data and methods suitability
Prediction
Combining train set and valid set for testing
PCA Method
Feature Selection
1.0, 0.9, 0.8, 0.7

Figure 1: Study procedure diagram

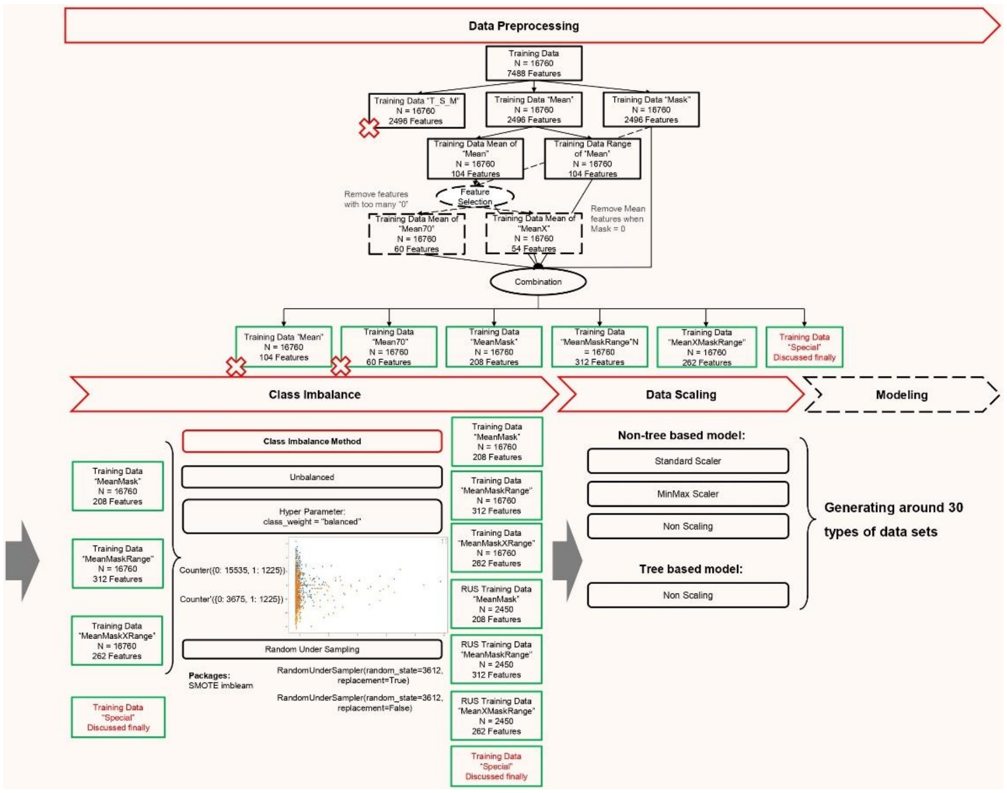
Data Preprocessing
Training Data N = 16760 7488 Features
Training Data "T_S_M" N = 16760 2496 Features
Training Data "Mean" N = 16760 2496 Features
Training Data "Mask" N = 16760 2496 Features
Training Data Mean of "Mean" N = 16760 104 Features
Training Data Range of "Mean" N = 16760 104 Features
Remove features with too many "0"
Feature Selection
Training Data Mean of "Mean70" N = 16760 60 Features
Training Data Mean of "MeanX" N = 16760 54 Features
Remove Mean features when Mask = 0
Combination
Training Data "Mean" N = 16760 104 Features
Training Data "Mean70" N = 16760 60 Features
Training Data "MeanMask" N = 16760 208 Features
Training Data "MeanMaskRange"N = 16760 312 Features
Training Data "MeanXMaskRange" N = 16760 262 Features
Training Data "Special" Discussed finally
Class Imbalance
Data Scaling
Modeling
Training Data "MeanMask" N = 16760 208 Features
Training Data "MeanMaskRange" N = 16760 312 Features
Training Data "MeanMaskXRange" N = 16760 262 Features
Training Data "Special" Discussed finally
Class Imbalance Method
Unbalanced
Hyper Parameter: class_weight = "balanced"
Counter({0: 15535, 1: 1225})
Counter'({0: 3675, 1: 1225})
Random Under Sampling
Packages: SMOTE imblearn
RandomUnderSampler(random_state=3612, replacement=True)
RandomUnderSampler(random_state=3612, replacement=False)
Training Data "MeanMask" N = 16760 208 Features
Training Data "MeanMaskRange" N = 16760 312 Features
Training Data "MeanMaskXRange" N = 16760 262 Features
RUS Training Data "MeanMask" N = 2450 208 Features
RUS Training Data "MeanMaskRange" N = 2450 312 Features
RUS Training Data "MeanXMaskRange" N = 2450 262 Features
Training Data "Special" Discussed finally
Non-tree based model:
Standard Scaler
MinMax Scaler
Non Scaling
Tree based model:
Non Scaling
Generating around 30 types of data sets

Figure 2: Data processing procedure diagram

2.2.1 Data preprocessing

First, the training, validation, and test sets of the independent variable data are divided into three sub-parts according to "Mean," "Mask," and "Time_Since_Measured," and then the "Mean" data are divided into 24-hour data by mean and extreme (named range in the dataset).

The next step for the obtained data using two feature engineering methods to select the independent variables that may be valuable: 1) the columns containing more 0s in the "Mean" set are directly deleted, and after trials, it is found that 70% is an excellent boundary to reckon, and finally from 104 columns down to 60 columns; 2) the data corresponding to the data with Mask of 0 are specially processed, and finally 50 columns are deleted in the "Mean" set. It is mentioned in the data analysis that the data corresponding to the "Mean" columns with Mask = 0 are processed, and in such a way exists the issue that the data with Mask = 1 are used multiple times, thus affecting the weight of the subsequent processing of the "Mean" set using the average again. Therefore, in the second method, the mean values in the "Mean" set are replaced by 0 if the corresponding Mask = 0, which can be shown in the function: c $Mean = \begin{cases} Mean, \, if \; mask = 1 \\ 0, \, if \; mask = 0 \end{cases}$. Then, the processed mean columns of "Mean" are margined to obtain a new Data frame, with a function $mean\ of\ Mean\ =\ \frac{mean\ Of\ Mean \times 24}{sum\ of\ Mask}$ to get the actual mean, because the number of non-zero mean values might be changed after the previous step. Eventually, columns with more than 40% (determined by trials) 0’s are dropped to get the final 54 columns.

In the last step of Data preprocessing, the above-mentioned data sets are combined for subsequent use and testing. The two data sets for the feature engineering mentioned above are named "Mean70" and "MeanX" respectively. During the checking stage, the datasets "Time_Since_Measured", "Mean" and "Mean70" training sets don’t perform well and are not considered in the follow-up study. The “Time_Since_Measured” values come from the last observation time of the “Mean” values, which may not have enough potential causality to the “Mean” value, and will be ignored currently. In addition, the "Mean" and "Mean70" sets both don’t consider the effect of “Mask” values on the model training; the possible reason for this is that there are too many filled data in the "Mean" value, so whether using all the "Mean" or deleting the columns with many zeros in the "Mean" sets cannot avoid this problem, which also confirms the significance Mask values. So, it can be considered to combine “Mask” with “Mean” data or filter the "Mean" data according to Mask. The other datasets are then combined separately to provide the final four datasets that will be passed to the next stage (with the Special data coming from the new data provided during the XGBoost method in Part 5, which will be presented later).

2.2.2 Class Imbalance

Recall the imbalance problem mentioned in the Data analysis part, most algorithms focus more on classifying the main samples and ignore or misclassify the few samples, however, the few samples are those that occur rarely but are important, so they can pose a bigger problem for data mining (Longadge & Dongre, 2013). Currently, there are three solutions to the imbalance problem: Resampling, boosting algorithm, and Feature Selection, and the Resampling method is used in this study because of its straightforward and efficient attribute. After that, a total of three class Imbalance methods are implemented, namely Unbalanced, using Hyper Parameter: class_weight = "balanced" in the function, and Random Under Sampling. For the second method, after several attempts, it is found that the 3:1 balance treatment is a better proportion choice, since if the ratio decreases more, it may lead to too much data being removed, making the results no longer accurate. At the end of this implementation, 15535 data in X_train set are reduced to 3675. For the Random Under Sampling, a package called “SMOTE, imblearn” is imported to the environment to apply “RandomUnderSampler” function, which randomly removes the majority class samples to reach a balance distribution. By the Class Imbalance method, more data sets are provided for further consideration.

2.2.3 Data Scaling

For algorithms that are not tree-based, most of these algorithms are Gradient Descent Algorithms and Distance-Based Algorithms, and data scaling is a very important means to improve their accuracy. This is due to the fact that different variables have different scales and thus contribute differently to the distance calculation, and the gradient descent has a different percentage of each approach. For example, min-max normalization in the KNN algorithm can provide good results, while SVM with a linear kernel can further improve the accuracy by standardization (Ambarwari et al., 2020). For the Tree-based method, the use of data scaling does not improve the correct rate because the proportional change of variables does not significantly affect the computation of the Tree. Other studies have tested XGBoost, Random Forest, GradientBoost, AdaBoost, and other methods and found basically no significant change in the accuracy (Ahsan et al., 2021). Therefore, in the study, Standard Scaling, MinMax Scaling, and Non-Scaling will be used for Non-Tree based method, while for Tree based method, the model will be trained directly using the data without scaling.

3 Modeling

3.1 Task 1

3.1.1 Model Selection

In total, Task1 uses different training sets for the initial training of a total of 14 models with simple loop-based tuning to select the models that may be used for subsequent in-depth analysis. By checking the performance of these 14 models based on ROC_ACU score, there are 5 models stand out, which are Logistic Regression, MLP Classifier, XGBoost, HistGradientBoost, and GradientBoost, where the first two are Non-tree based models, and the last three are Tree based models. Looking back at the best dataset for which all Non-Tree based models are applicable, it can be found that the selected data processing methods are mmx as well as no processing, reflecting that the dataset is actually not suitable for Standard Scaler method.

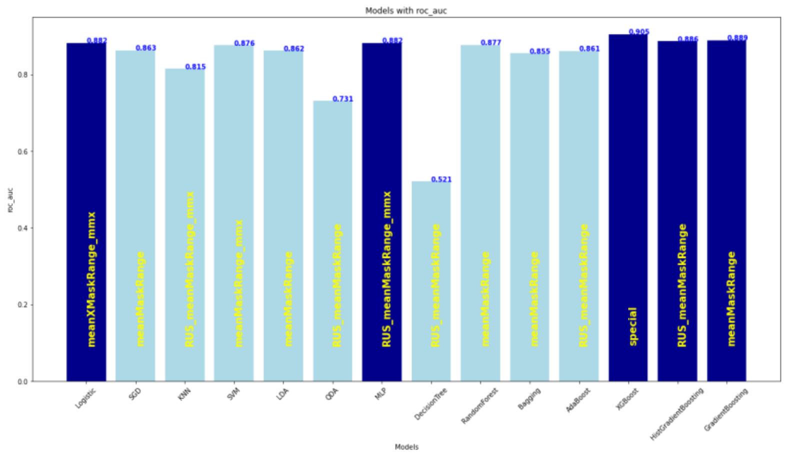

Figure 3: Task 1 classification study, training performance results from different models

3.1.2 Model Diagnostics and Discussion

For the Non-Tree based models, Logistic Regression is chosen to be the best performed model, as it has the highest score as MLP Classifier and also with strong interpretability. In addition, during the parameter tuning trials for MLP Classifier, the layer size is the most important parameter which is significantly correlated with the ROC_AUC score. However, the effect of layer adjustment on the result is very unstable, for example, reducing the second layer of two layers from 200 to 199 can make the final score fluctuate greatly, the logic of which is difficult to be discovered. Eventually, Logistic Regression model is chosen to be the representative of the highest interpretable model. For the Tree based models, since they both perform less well than XGBoost and have worse explanations than logistic regression, the final choice is the XGBoost, which will be explained in detail later. It is worth briefly explaining that the decision tree algorithm performs very poorly, probably due to 1) the high number of unrelated variables, which affects the results of the decision tree in finding variables' interactions; 2) each split of the tree will reduce the data set and intentionally creating splits will introduce additional bias; 3) the algorithm has extreme variance and doesn't optimize for this based on residuals as Boost would.

In this section, the Logistic Regression model and Quadratic discriminant analysis are chosen to make a comparison of their performance, which is capable of providing insights for other Non-Tree based models discarding. In two-dimensional coordinates, logical regression is classified by a straight line, which shows that the essence of logical regression is linear classification. This method uses statistics, pattern recognition, and machine learning methods to try to find a linear combination of the characteristics of two types of objects or events, to be able to characterize or distinguish them. The combination obtained can be used as a linear classifier, or, more commonly, to reduce the dimension for subsequent classification, which comes to the similarity with the widely used model LDA in biomedical research, face recognition, etc. Quadratic discriminant analysis (QDA) is a variant of LDA, in which a single covariance matrix is estimated for each type of observation. QDA is particularly useful if it is known in advance that individual categories exhibit different covariances. The disadvantage of QDA is that it cannot be used as a dimension-reduction technology.

After fitting the model to task 1, the average score (ROC_AUC) for Logistic Regression and Linear discriminant analysis are about 0.85 and 0.86, respectively, but for Quadratic discriminant analysis, the average score is only around 0.7, and it will drop to 0.5 if we did not apply the random under-sampling method to the training dataset. We can conclude that the mortality prediction is linearly separable, so the linear classifier like Logistic regression and LDA can fit the dataset better.

3.1.3 Parameter Tuning

For Logistic regression, we focus on the performance of tunning different hyperparameters by Grid Search CV Algorithm to provide a more valid result than simple loop-based parameter tuning, including penalty, solver, l1-ratio, tolerance, and maximum iteration. The best performed and finally selected parameters are: (C = 1, random_state=0, solver='liblinear', penalty = 'l1').

In figure 3.1, we can see that with a higher C value, the score of models will generally increase, but l2 logistic model does not perform well compared to l1 logistic model when C is large. The reason for this may come from the fact that the screening of Feature by the regularization parameters decreases along with the increase of C, while the penalty of l1 can make the coefficient of some parameters really reach 0, able to provide a similar effect as C. Since different solvers support unique penalty methods, there is an interaction effect between C value and the choice of solver for the logistic model. After parameter debugging, we notice that for this dataset, the l1 penalty performs better than l2 and elasticnet, so we choose the solvers that provide the l1 penalty for subsequent tuning. Figure 4.2 shows that there is almost no difference between solvers who provide l1 penalties while tunning the C. In figure 4.3, it can be found that with a larger value in C, the model score will perform better than the model with smaller values in C. Looking at Figures 4.4 and 4.5, there is no effect of maximum iteration and tolerance value on the performance score, which means that for the selected dataset, logistic regression does not require excessively deep iterations to achieve optimal results. Eventually, the solver of “liblinear” is the best solution for the choice of logistic regression.

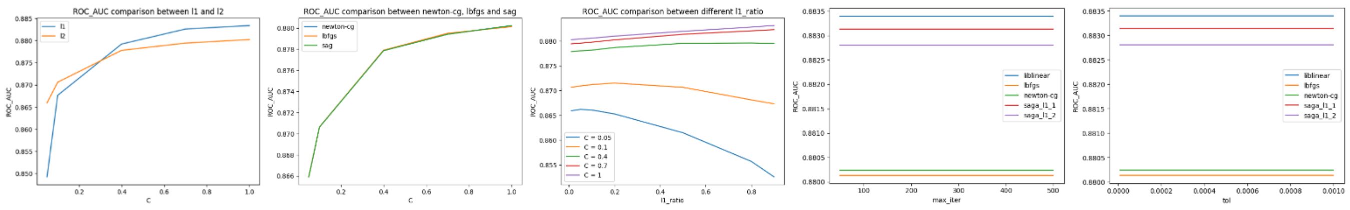

Figure 4: Logistics Regression Performance with different Parameters (named 4.1, 4.2, 4.3, 4.4, 4.5, respectively)

3.2 Task 2

3.2.1 Model Selection

Plenty of machine learning models were attempted and assessed by root mean square error (RMSE) in the regression task. As a result, Random Forest (RF) stands out using the optimized hyperparameter. Our parameter selection for RF was based on random research that goes through possible candidate parameters and assesses model performance with relevant parameters, which can reduce the search time while ensuring certain model accuracy. The random search algorithm randomly selects a value in an expected range for the hyperparameters, performs model training, and verifies the model to find the best-performing set of hyperparameters among all random combinations. Following the random search, Grid Search then took advantage of permutations and combinations among the possible parameters taken from Random Search. As a result, our best performing Random Forest model parameters were found: (max_features=6, n_estimators=740, min_samples_leaf=1, min_samples_split=3, max_depth=46, bootstrap=False). While the best performing parameters for Random Forest and other algorithms were found, we failed to complete Grid Search for SVM due to too many permutations and combinations and slow overall processing time in SVMs for large and high dimensional data.

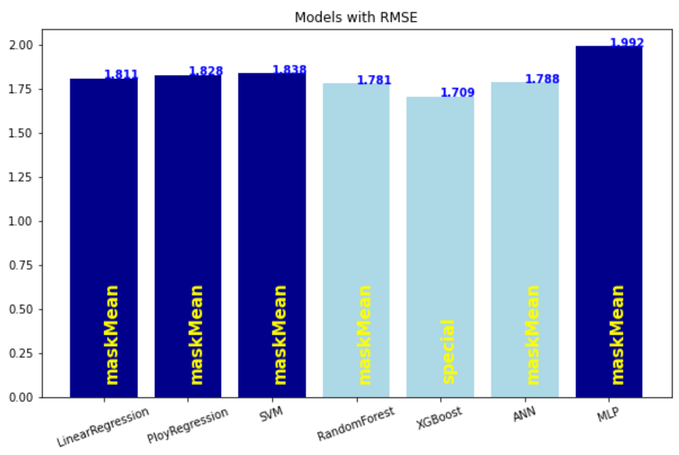

Figure 5: Task 2 quantitative prediction study, training performance results from different models

3.2.2 Model Diagnostics and Discussion

The performance of some models, such as SVM, has been unsatisfactory for the regression task. We assume there are several reasons for this: (i) SVMs are not suitable for large datasets - Although the SVM implementation has a well-founded theoretical basis, it is not suitable for analysis for relatively large data sets since the complexity of algorithm depends significantly on the size of the data set. That is, the training time grows fastly with the size of a dataset, which becomes untrainable and unusable due to computational constraints (Boser, Guyon & Vapnik, 1992). In our project, we could not locate the best parameters for SVM due to time and computational constraints. (ii) SVMs' unsatisfactory performance in imbalanced datasets: (i) the weakness of soft margin optimization problem - Hyperplane bias towards the minority class when trained with imbalanced data, and (ii) imbalanced support vector ratio - the ratio between negative and positive support vectors is imbalanced, leading to the case that the data points near the hyperplane decision boundary tend to be classified as negative support vectors (Batuwita & Palade, 2013). We assume that the introduction to class weights can address this issue, so the magnitude of the positive support vectors will be higher than that of the negative support vectors. Except for SVM, the multilayer perceptron (MLP) model-a classic type of deep neural network- also yields one of the lowest RMSE in the regression task. One possible cause is the amount of training data used. The deep neural network consists of many neurons as processing units, bringing massive degrees of freedom. In this case, quite a lot of iterations are needed to find the optimized hyperparameters. Insufficient data size would lead to overfitting during this process. Another reason is the difficulty in determining the proper architecture for ANN. The factors, including the batch size, learning rate, activation function, and number of hidden units, would significantly affect the performance. Therefore, it is easy to choose the wrong architecture and get stuck at the local minima.

3.2.3 Parameter Tuning

First, we directly examine the feature importance of predictors to find which factor is significant in predicting the patients' length-of-stay (LOS) based on the optimization. As we can see from the figure, the positive end-expiratory pressure (PEEP) set appears to be the most significant predictor, followed by tidal volume observed, respiratory rate, respiratory rate set, tidal volume set, blood urea nitrogen, oxygen saturation, heart rate, peak inspiratory pressure and partial thromboplastin time. As the differences between each predictor are minor, they also need to be considered in predicting LOS. Regarding how the PEEP set affects the LOS of patients, the topic is too general to research. However, Mercat et al. (2008) discovered that the strategy in setting PEEP did reduce the duration of mechanical ventilation and the duration of organ failure in terms of lung injury, which is consistent with our findings. As for tidal volume observation and set, most research is also limited to lung disease. The research conducted by Needham et al. (2015) stated that timely adherence to setting tidal volume ventilation is crucial for improving the survival of ARDS patients whose lungs cannot provide the body's vital organs with enough oxygen. The gap in the research fields regarding the effects of these significant factors on patients' LOS may become the direction of future research.

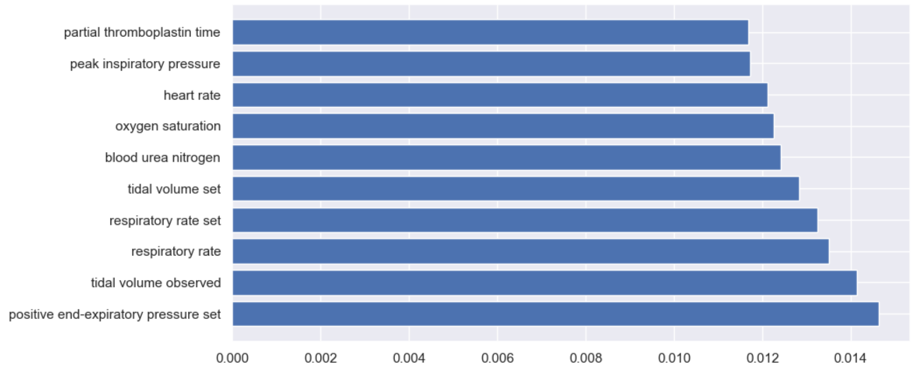

Figure 6: Feature importance comparison in Task 2

After investigating the importance of each factor, we reduce the number of depths to four and visualize the resulting Random Forest regressor for illustration. As we can see from the graph, there are scatterplots of the features used for the split. The horizontal dashed lines indicate the target's mean for the left and suitable buckets in decision nodes, and the vertical lines are the split point. For instance, the topmost node splits all data points into two groups based on whether the peak inspiratory pressure is more significant than 0.26. If it is larger than 0.26, we may predict the LOS of patients using the mean, which is approximately 3. This is useful when the complete information of a particular patient is absent. It is worth noting that the node would differ depending on the choice of depth for visualization. Therefore, the situation for the optimized RF model with 46 depths might be different.

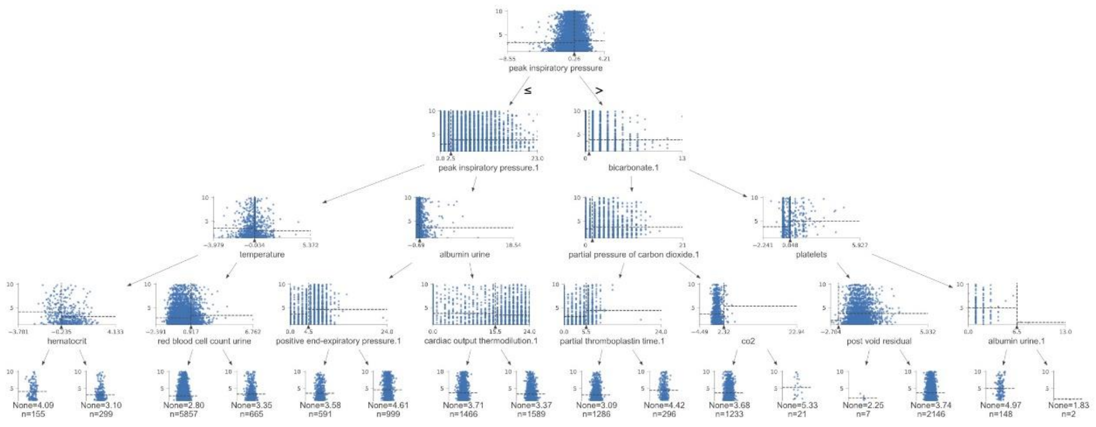

Figure 7: Four depths Random Forest Features Diagram

4 Best performance: XGBoost

4.1 Reflection on feature selection

When reflecting on our feature selection strategy, it is too subjective to just combine all the 7844 columns with the same header belonging to different times. Meanwhile, even if we do not find the best strategy to utilize the data from "Time_Since_Measured", it does not prove any uselessness. Despite choosing the features manually, we explored choosing by the algorithm. There are some common methods to do so: 1) Select K Best supported by sci-kit learn library; 2) Forward and backward selection supported by mlxtend library; 3) Compare the coefficients of linear regression; 4) Random Forest and XGBoost with attribute feature_importances.

Depending on the strategy before, we use balanced data to compare these methods.

4.1.1 Select K Best

Select K Best is based on the F-ratio of the ANOVA table. Firstly, we fit the algorithm on a balanced dataset with the ratio 3:1

of ‘0’ and ‘1’. Secondly, we take look at the feature scores. We get the features whose score is larger than the mean of scores. It shows the significance of “Time_Since_Measured”. Lastly, we use these features to fit a logistic regression model. However, the ROC_AUC score only attains 0.69.

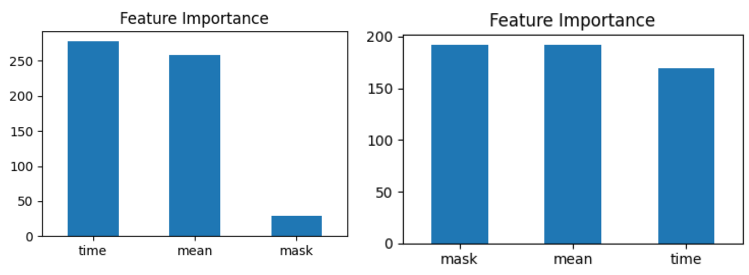

Figure 8: Linear Coef_ (Left) and K Best (Right)

4.1.2 Forward and Backward Selection

By intuition, it should be an excellent method to solve this problem. However, in actual practice, even choosing 10 features with forward selection method costing more than 15 minutes while performing poorly on validation. Therefore, this method is aborted.

4.1.3 Linear Regression Coefficients

Fit the whole balanced model with linear regression, generating 565 features whose absolute coefficient is larger than 0.4. Then we use the reduced balanced dataset to train and validate the score. A simple logistic regression attained ROC_AUC score of 0.81. Meanwhile, it also shows the significance of “Time_Since_Measured” (See *Figure 5 Linear Coef_(Left)*).

4.1.4 Random Forest and XGBoost

By now, we still do not get a model that outperforms the model fitted on manually selected features. By researching the sci-kit library, only Random Forest of ensemble learning classifiers has the attribute feature importance. Noting that our motivation is to select features that do not fit the model on the whole dataset while it may produce excellent results by classifiers like Hist Gradient Boosting. Therefore, another boosting is XGBoost, which also has the feature importance attribute. Figure 6 below shows the ROC_AUC score of XGBoost, Random Forest, and Hist Gradient Boosting compared with different ratios of class imbalance (all of them are untuned). It performs exceptionally well with all the features compared with manually selected features before. Therefore, we choose to move on with Random Forest and XGBoost to choose features.

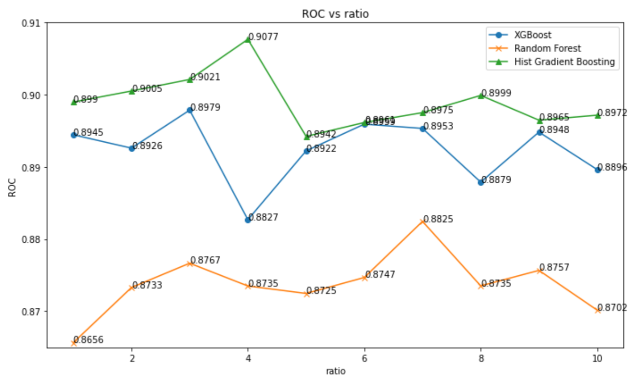

Figure 9: ROC_AUC vs ratios (XGBoost, Random Forest, Hist GDBoost trained with All Features)

4.2 Reduce the features

From figure 6, we will keep using our strategy of random under sampling for Random Forest, and XGBoost with ratios of 1:7 and 1:3, respectively. Our steps are: fit the model on the whole balanced dataset, get the feature whose importance is more prominent than importance; fit the model on reduced once balanced dataset, get the feature whose importance is largthe er than mean of importance; fit the model on reduced twice dataset, tune the model on this dataset.

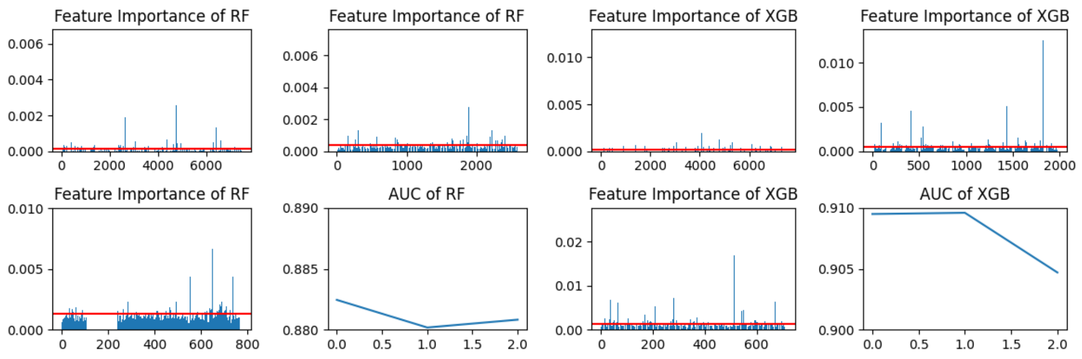

Figure 10: Random Forest and XGBoost comparison on Feature importance and AUC

We conclude that XGBoosting is much better than random forest, and each method produces around 700 features in the end. Continuing this recursive reduction produces worse results. Meanwhile, the test scores of XGBoost in each reduction are 0.918, 0.919, and 0.922, respectively.

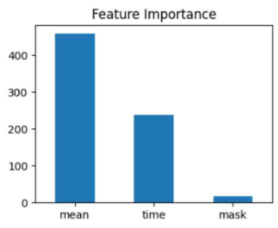

Figure 11: Feature importance at the end of feature engineering

Also, we noted that the “time” is very important, we cannot delete it. When adding the “time” into features, the importance of the “mask” decreases. Compared model abandoning “time”, it is better without numerous “mask” data. It indicates the existence of a correlation between “time” and “mask”.

4.3 Advantages of XGBoost

Besides the performance of XGBoost, XGB supports GPU speeding. When testing on the whole dataset, XGB(CPU), XGB(GPU), Random Forest, and Hist Gradient Boosting cost 7 minutes, 20 seconds, 1.5 minutes, and 10 minutes respectively. It enables us to find the process of tuning parameters. The capability of multi-tasking. When shifting to the regression model, the same reduction strategy performs well, which produces RMSE from 1.72 to 1.73 for validation.

5 Limitation

During the data processing and model training procedure, some existing problems are not yet figured out, which will be introduced here for further reflection. In the data scaling process, two significant methods are implemented: Standard Scaling and Min Max Scaling. However, due to the time constraint, it is hard to figure out whether some other scaling attempts, such as MaxAbs (MA), Robust Scaler (RS), and Quantile Transformer (QT), are more efficient in data processing (Ahsan et al., 2021), which may bring a potential increase in the final prediction. As mentioned earlier, one possible reason for unsatisfactory performance of MLP is the difficulty in determining the appropriate architecture for the neural network. Based on Bashiri and Farshbaf Geranmayeh (2011), tuning the parameters is one of the biggest problems using ANN since there is no specific method to choose optimal parameters. Our study using MLP is also stuck on finding the proper structure of the neural network. For instance, the number of hidden layers and the number of neurons in each layer are found to be challenging to determine. We barely make decisions based on past experiences. However, optimized features vary under different applications. Another limitation is that a lot of time is wasted on Grid Search as the dataset is relatively large. Except for timing-consuming, Grid Search faces the danger of overfitting. Although we manage to combine Random Search CV and Grid Search CV to reduce the training time, hyperparameter tuning using GPU could be the better alternative.

6 Conclusion

Based on our research, we first confirmed the high value of data processing for model training and tried many possible approaches for this purpose. In the Data pre-processing phase, we found that the Mask variable significantly impacted the findings and also gained insight into the positive effect of both the mean and the extreme deviation of the data on the model training. These findings can be confirmed by analysis and literature review. In the class imbalance section, the data imbalance problem is explored, and both hyperparameters and random_under_sample are tried for adjustment, which can contribute to the model improvement. In the data scaling section, Standard Scaling and MMX Scaling treatments are used to fit Non-Tree based models better, and the best data sets for logistic regression are derived from this treatment. In task 1, a total of fourteen models were considered, and an initial selection was made, and eventually, five models stood out for their accuracy. We finally selected logistic regression, which has the best explanatory power, and XGBoost, which has the best prediction performance. We explored many possible problems with the MLP Classifier and Decision Tree methods, such as difficulty in confirming the tuning direction and unstable results. Last but not least, the most critical parameters in the selection of logistic regression are C and penalty. In task 2, we applied six machine learning models, including linear regression, polynomial regression, Random Forest, SVM, MLP, and XGBoosting, with different data scaling approaches. It turns out that the XGBoosting yields the most accurate prediction. By examining the feature importance, the top ten factors are positive end-expiratory pressure (PEEP) set, tidal volume observed, respiratory rate, respiratory rate set, tidal volume set, blood urea nitrogen, oxygen saturation, heart rate, peak inspiratory pressure and partial thromboplastin time. Besides, the excellent interpretability of Random Forest allows us to make predictions without complete information about patients. Finally, we focused on enhancing the XGBoost method by adding the "Time_Since_Measure" data, which we had not considered before, and by processing the data in plenty of feature selection ways. The importance of features was detected and visualized in each reduction phase. We finally achieved the best prediction results from this data with more than 700 features and achieved a good balance between runtime and prediction accuracy. At the same time, the Feature selected by the method is applied to both Task 1 and Task 2 predictions, demonstrating the conclusion of multi-tasking learning. Limitations in the study and points to consider for further improvement are also presented in the final part.

Reference

Ahsan, M. M., Mahmud, M. A. P., Saha, P. K., Gupta, K. D., & Siddique, Z. (2021). Effect of Data Scaling Methods on Machine Learning Algorithms and Model Performance. *Technologies, 9*(3), 52.

Ambarwari, A., Adrian, Q.J., Herdiyeni, Y. (2020). Analysis of the Effect of Data Scaling on the Performance of the Machine Learning Algorithm for Plant Identification. *J. Resti, 4*(1), 117-122.

Batuwita, R., & Palade, V. (2013). Class Imbalance Learning Methods for Support Vector Machines. *Imbalanced Learning*, 83–99. https://doi.org/10.1002/9781118646106.ch5

Bashiri, M., & Farshbaf Geranmayeh, A. (2011). Tuning the parameters of an artificial neural network using central composite design and genetic algorithm. *Scientia Iranica, 18*(6), 1600–1608.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory - COLT '92*, 114-152.

Caner, M., & Hansen, B. E. (2004). INSTRUMENTAL VARIABLE ESTIMATION OF A THRESHOLD MODEL. *Econometric Theory, 20*(05).

Chrusciel, J., Girardon, F., Roquette, L., Laplanche, D., Duclos, A., & Sanchez, S. (2021). The prediction of hospital length of stay using unstructured data. *BMC Medical Informatics and Decision Making, 21*(1).

length of stay. (n.d.) *Segen's Medical Dictionary.* (2011). Retrieved December 6 2022 from https://medical-dictionary.thefreedictionary.com/length+of+stay

Longadge, R., & Dongre, S. (2013). Class Imbalance Problem in Data Mining Review. ArXiv:1305.1707 *[Cs], 2*(1).

Mercat, A., Richard, J.-C. M., Vielle, B., Jaber, S., Osman, D., Diehl, J.-L., Lefrant, J.-Y., Prat, G., Richecoeur, J., Nieszkowska, A., Gervais, C., Baudot, J., Bouadma, L., Brochard, L., & Expiratory Pressure (Express) Study Group. (2008). Positive end-expiratory pressure setting in adults with acute lung injury and acute respiratory distress syndrome: a randomized controlled trial. *JAMA, 299(6)*, 646–655.

Needham, D. M., Yang, T., Dinglas, V. D., Mendez-Tellez, P. A., Shanholtz, C., Sevransky, J. E., Brower, R. G., Pronovost, P. J., & Colantuoni, E. (2015). Timing of Low Tidal Volume Ventilation and Intensive Care Unit Mortality in Acute Respiratory Distress Syndrome. A Prospective Cohort Study. *American Journal of Respiratory and Critical Care Medicine, 191(2)*, 177–185.

Thompson, G., O'Horo, J. C., Pickering, B. W., & Herasevich, V. (2015). Impact of the Electronic Medical Record on Mortality, Length of Stay, and Cost in the Hospital and ICU. *Critical Care Medicine, 43(6)*, 1276–1282.