STAT3612 Final Report

Author: LOU HUAJIE, LI HANWEN, WU JIAXIN, XUE LUHAO

December 7, 2022

Contents

Abstract

As electronic health records are helpful in clinic prescription and allocation of medical resources, our group worked on a project that adopted machine learning methods to predict mortality status and staying length. With data pre-processing methods that reduce the dimension of data substantially, we proposed multiple baseline models for two tasks respectively. After fine-tuning hyper-parameters, it turned out that tree algorithms perform better among all experimented models. Specifically, the final models are XGboosting for Task 1 and a linear combination of three models for Task 2.

1. Introduction

"Machine learning" has gained unprecedented publicity over recent years. As a state-of-the-art technology, it refers to methods that leverage data to improve the performance of computers on specific tasks, which emulates the learning process of humans. Among all the goals machine learning algorithms seek to achieve, classification and regression are of enormous significance. Classification aims at assigning a set of data with class labels, while regression predicts continuous responses based on predictors. In this paper, we attempt to deploy learning algorithms such as linear regression, generative additive model, random forest, support vector machine, and neural network to predict patients' mortality status and length of staying in ICU based on the MIMIC-III dataset. After the implementation of all proposed models, comparison and analysis adopting various metrics will be made.

2. Related Work

With the introduction of MIMIC datasets, early works on mortality and LoS predictions mainly adopted some simple scoring systems including acute physiology and chronic health evaluation II (APACHE II), simplified acute physiology score II (SAP II), and sequential organ failure assessment (SOFA) [8, 10]. Meanwhile, other researchers took some traditional machine learning methods combined with appropriate feature engineering [10]. Desirable results and benchmarks have been proposed using support vector machines (SVM) and random forests for both classification and prediction tasks. These experiments largely focused on prior medical-related domain knowledge and complex feature selection procedure. With the boost of deep learning, recent works have put more attention to using neural networks to automatically select features and perform the prediction both at once [3, 10, 14]. Regarding the mortality and LoS predictions, our work will basically focus on comparisons of different machine learning models and improvements of prediction results.

3. Data Processing

3.1. Dataset Introduction

Each patient sample in the dataset has 104 features with 72 variables describing the feature. The first 24 variables represent masks, which is 1 if the value is present at this timestep and 0 otherwise [14]. The next 24 variables are the imputed values of the feature, and the last 24 variables represent the time since the last measurement of this feature. Since each x has more than 7000 values and many columns are highly correlated, we need to pre-process the data and select appropriate features before model training.

3.2. Data Pre-processing and Feature selection

Since the columns in our dataset are large compared to the data size, it will make many models not robust and overfit to the training dataset. Thus, we conducted some feature selection methods in order to reduce the feature dimension.

3.2.1. Previous Work

As mentioned in related work by others, some simple scoring has already given an insight into the status of patients staying in the ICU. Given measured values of some particular medical indicators, APACHE II, SAPS II, and SOFA scoring systems could be regarded as an elementary overall prediction for ICU subjects. Regarding the structure of our dataset, the mean values of 104 medical indicators have been recorded during the first 24 hours, which directly fits the scope of APACHE II and SAPS II. Also, these two models have been shown to deliver good specificity and sensitivity results when the first 24 hours of ICU admission are available [8].

However, since the given MIMIC-Extract dataset only contains normalized values instead of raw data, it is not applicable for us to directly calculate these two scores. Instead, we looked deeper into the components of APACHE II and SAPS II and fetched these indicators from our dataset if they were measured in the MIMIC-Extract. Meanwhile, Purushotham [10] used Feature Set A, a revised version of the SAPS II indicator, as one feature set to build benchmarks in ICU-related predictions. We then simply took the union of the measured indicators in Feature Set A and previous indicators obtained in APACHE II and SAPS II. This gave us our finalized version of the features, FS-1 containing 24 features, which can be shown in Appendix A.

3.2.2. Dimension Reduction Methods

As the famous curse of dimensionality states, increasing the feature quantity will result in a model's lousy performance. A high dimension not only means a high computational cost but often can lead to the overfitting of the model. The data with high dimensions also tend to be highly correlated. Hence to obtain a good-performing model, the reduction of feature dimension is a necessity. We will introduce four main methods to do feature reduction.

1. **Conventional Principal Component Analysis (PCA)**
   Conventional Principal Component Analysis (PCA) is a helpful dimension reduction technique. It transforms the attributes of a dataset into an uncorrelated dataset [5]. It can reduce the dimension of features and maintain the variability of the data at the same time. Although PCA is widely used in machine learning and can significantly help unsupervised learning, it performs poorly in our classification project [9]. Details will be shown in Section 5.

2. **Function: mutual_info_classif**
   This function is inside the sklearn.feature_selection package. It will calculate the mutual information of every feature and the response variable before we fit any model to it. Mutual information can be regarded as a similar tool to the correlation between two quantities. In the mortality prediction task, we first ordered all the 7488 features based on their mutual information scores and fetched the previous few important features to be included in our models. We chose the best number of selected features by the grid search method. When we used more features, the grading metric, say the AUROC, on the training and validation datasets will decrease monotonically. Up to some thresholds of chosen features, the results on the validation datasets will increase, which represents a signal of overfitting. Thus, we monitored the AUROC on validation datasets in order to find the most desirable models. In particular, in the tree-based methods such as boosting, we set an early stopping iteration of 15 rounds in order to encounter the overfitting problem.

3. **Function: feature_importances_**
   Here we use a particular model, like random-forest, to fit all the variables and then utilize the feature_importances_ function to check their importance. Record all the importance in one list and set an appropriate bar to select the features. Here we can apply the grid search method to find the most efficient bar. The features selected by this method are according to the model's performance on the validation set. So detailed information will be provided together with the models.

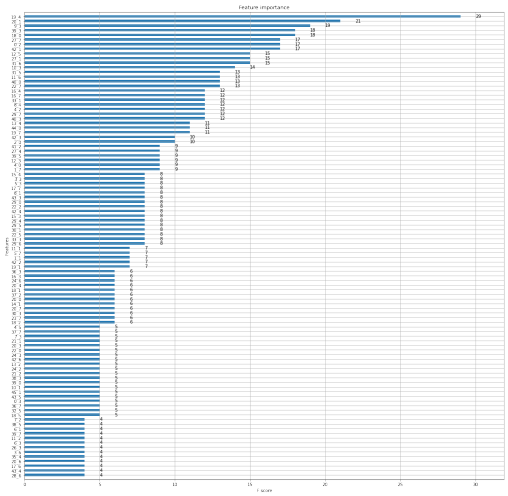

Figure 1. Comparison of importance across selected features

4. **Literature**
   We consult the relevant literature to derive the list of

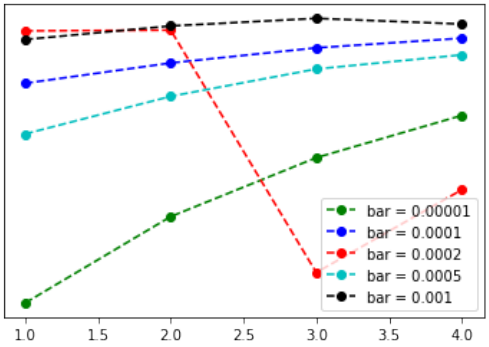

Figure 2. The accuracy changes with different depths and importance bar

the essential related factors. For example, Johannes et al. (2009) reveal that 'Blood Urea Nitrogen' plays a significant role in predicting the mortality state of a patient [15]. Here we append the feature 'Blood Urea Nitrogen' to our feature list. With some more consultation, we obtained the one possible version of the feature list which might be used in our model. It will be shown in Appendix B.

3.2.3. Aggregation of Time-series Data

In this part, we performed feature engineering and splitting. For some features, such as creatinine ascites, only a few patients have been measured during the time period, so the imputed values for these features are set to the global mean of the training data, which is 0 after normalization. With a lack of observed values in the training dataset, models including these features may be less comprehensive. Therefore, if more than half of the patients in the training dataset have no observed value for a feature, the feature will be excluded from our training model.

For each model, we tried many versions of datasets. We always started with the whole dataset and performed the feature reduction methods mentioned above. Then we would try some others. For example, since the values in mask and time_since_measured merely record time information of measurements and are not directly related to a patient's condition, we only used the imputed values of the features in our model. the final version of the dataset depends on the performance of the model on the validation set.

Through observation, the 24 imputed values of each component are largely repetitive, so we decided to take the mean, maximum, and minimum of the 24 values to represent a feature for each patient. Besides, the health condition of a patient may also be related to the variation amplitude of certain indexes. So we also considered the average variation of a feature, which is calculated by the difference between the first and the last measurement divided by the time in between. To sum up, we tended to abandon the usage of a total of 72 features if each medical indicator. Instead, we can simply take the sum, mean, range, and other simple statistics including but not limited to maximum, minimum, skewness, and variances of the mean, mask, and time_since_measured across the 24 hours. And in the final version of our second model, we partly use the sum of the mask, the mean of the 24 normalized values, and the range of the normalized values to represent the feature from the feature list.

3.2.4. Deal with Imbalanced Data

As the training set is imbalanced with more data in class 0, we tried to randomly make the training data in class 0 into sizes similar to class 1. We basically used random sampling methods including oversampling and undersampling. We found that naive oversampling outperforms undersampling since we always used fewer data in fitting models. However, random oversampling always has the overfitting issue since we would see the same data points from the minority class over and over again.

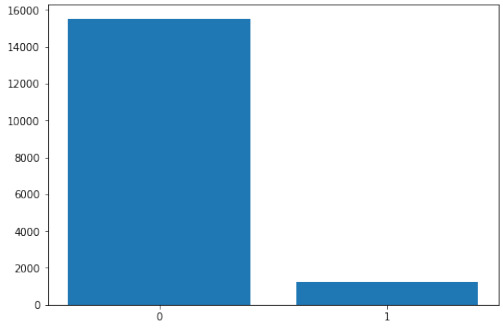

Figure 3. Number of training data in the two classes

New sampling techniques such as the Synthetic Minority Oversampling Technique (SMOTE) and the Adaptive Synthetic (ADASYN) have been developed [4]. After we tested the comparisons among different sampling methods on both training and validation datasets, we finally chose SMOTE as our random sampling method. SMOTE generates new samples of the minority class, class with label 1 in our task one, favoring interpolations near decision boundaries so we are less prone to overfit on data points that are easy to be classified. This tends to work the best in our dataset possibly because, in all 104 medical indicators, some ranges or fluctuations of particular indicators may be fatal and directly cause death in ICU. In lieu of straightforwardly copying existing samples, SMOTE oversamples new observations in order to tackle problems of overfitting and actually improve the classification results.

Besides, we could add more penalties for some wrong classifications of label 1. In other words, for instance,

we could reward one point for each correctly classified case of label 0 which is the majority class while we rewarded ten points for the correctly classified case of label 1. Fortunately, this can be automatically done by some well-developed models such as XGBoost. Furthermore, as a resolution to the overfitting problem, we performed k-fold cross-validation combined with the grid search on these training sets to find the best model.

4. Optimization

To measure the performance of our models, we employ the loss function to evaluate how well the models fit the data set and predict the outcome [7]. Additional penalty terms, such as ℓ_2 norm and ℓ_1 norm, will be added to the loss function to improve the metric and avoid over-fitting [13]. The main problem turns into the optimization problem of minimizing the improved loss function. SGD optimizer is of frequent use in logistic regression and neural networks. In our project, we will try mini-batch gradient descent to cut down on the variance of parameter updates in neural network models, resulting in more stable convergence [11]. To deal with the problems caused by the improper learning rate, such as slow convergence and divergence, Nesterov accelerated gradient (NAG) [11] and Newton-Raphson's method will be applied. We will also apply other methods like shuffling to improve our optimization algorithm [11].

5. Methodology

In this section, we will briefly introduce the models we adopted in mortality classification (task 1) and LoS prediction (task 2) tasks along with some feature selection methods we have tried.

5.1. Task 1: Classification

Regarding task 1, we are required to predict the mortality status of subjects staying in the ICU. Therefore, it will be a binary classification problem. We will introduce all the models we have adopted.

5.1.1. Logistic Regression

Since we need to predict the mortality status of objects, which is a binary number taking either 0 or 1, we started with the classical parametric classification model, the Logistic regression model (1). The model suggests a linear relationship between the logit and features. We would mainly use this model to test if there is any improvement when we used the data pre-processing methods we have introduced in Section 3 because this model takes a shorter time to fit and can be interpreted easily.

$$\hat{p} = \frac{exp(b_0 + b_1X_1 + b_2X_2 \dots b_pX_p)}{1 + exp(b_0 + b_1X_1 + b_2X_2 \dots b_pX_p)} \quad (1)$$

We set the vanilla Logistic model as the baseline model. Meanwhile, we attempted to improve the Logistic model with different feature sets, dimension reduction methods, aggregation of time-series data, and sampling methods. If the related features do not perform well in the baseline model, we will no longer consider using them in some more advanced models due to longer training and fine-tuning time, especially for tree-based methods.

We implemented logistic regression with the function in the sklearn library, and L2 regression with regularization strength 1e5 was chosen after comparisons of the model accuracy.

Logistic regression with different feature sets and feature aggregation

At the beginning of our project, we would like to find which feature sets fit our model the best by using the simplest Logistic regression models. In Section 3.2, we have briefly introduced some feature sets we attempted to improve the model performances. Firstly, we tried the FS-1 which contained some rather important medical indicators proposed by other papers. Also, instead of using the raw values for the 24 hours, we tried some aggregations of columns as introduced in Section 3.2.3. However, the AUROC results were not good probably because we did not cover enough information in the original dataset for the model to learn. We considered this dataset may be generalized well when the measured medical indicators were fewer and worked as an elementary criterion for mortality prediction.

Then, we used the list of features from the literature in Appendix B which actually used more features than in FS-1. Actually, it performs better in terms of AUROC. The performance worsened only a little when we aggregated the columns with time series. Nevertheless, with the information loss in the aggregation, the feature dimension will largely shrink which in turn can be interpreted much more easily.

Finally, we adopted the functions wrapped in Python packages. We extracted the first few dimensions with the highest mutual information scores before we fit the Logistic model. Actually, this method performed the best without any grid searching. When we only selected the first 600 features, the model was able to achieve an AUROC of about 0.84 on the validation dataset. This may be caused by the fact that the principle of this method is based on the correlation between the features and the response so that it can best fit this dataset. An example of choosing the first 600 out of 7488 features that work as an example of comparisons when using different data processing techniques.

Logistic Regression with Dimension Reduction

To improve our logistic regression model, we tried to

combine dimension reduction methods with logistic regression.

Firstly, we used Principal Component Analysis (PCA). PCA is a popular multivariate technique for compressing a large dataset with many inter-correlated dependent variables. It extracts the most important information from a dataset and increases data interpretability by transforming the data into a new set of principal components [1].

Secondly, Truncated Singular Value Decomposition (SVD) was applied. Unlike PCA, this method does not center the data before calculating the singular value decomposition. This means that it can efficiently handle sparse matrices, which fits the scope of our dataset well.

Lastly, the autoencoder was combined with our logistic regression model. We have used deep neural networks to encode the features with the bottleneck layer of 64, 128, and 256 neurons respectively. Dropout layers are included to avoid overfitting. However, in spite of a decrease in the reconstruction loss for the autoencoder as the number of neurons encoded in the hidden layer, the model fitting to the Logistic regression model actually does not improve. The results of using 64 neurons in the bottleneck layer and other dimension reduction methods are shown in Figure 1.

Logistic Regression with Data Resampling

Since the original dataset is imbalanced with most patients belonging to class 0, as mentioned in Section 3.2.4, we decided to resample our dataset for the logistic regression using Synthetic Minority Oversampling Technique (SMOTE).

5.1.2. Discriminant Analysis

Linear Discriminant Analysis (LDA) has a decision rule depending only on features through linear combinations. This model assumes observations within every class follow a normal distribution and share the same covariance matrix [6]. If the processed features are linearly related to the output and the dimension of features is much less than the sample size (i.e., $p \ll n$), LDA will perform well in terms of accuracy.

We applied LDA with the function in sklearn library using the least square solution solver and auto shrinkage. Overfitting to the training data was encountered, probably due to the large dimension of features.

5.1.3. KNN

As the first non-parametric method we include in our project, KNN, an example-based learning approach, finds the k nearest samples in the training dataset based on some distance metrics (Euclidean distance used in our case). It does not assume any particular requirements on the input features (i.e., normality). This method enjoys high accuracy and is not sensitive to outliers. However, we have more than 16K training samples, we have to calculate the distance value for each data in the dataset, which can be very time-consuming in practice. Furthermore, the training dataset size is large, which makes the space complexity high. This method does not give any information about the underlying structure of the data, so we cannot explain the effect of features on the classification results. However, we would still adopt this method by trying different values of K to see the performance in practice.

5.1.4. Random Forests and Boosting

Since tree-based methods are commonly used in both classification and regression tasks. The details will be illustrated in Section 5.2.2. In practice, tree-based methods always need longer time and more hyperparameters to fine-tune. Grid searching is again used in our pipeline. In order to get better prediction results, we mainly tune the hyperparameters, including max_features, n_estimators, and min_sample_leaf. The argument max_features represents the maximum number of features that will be used in each fitting tree. Increasing max_features generally improves the performance of the model because at each node we have more choices to consider. However, this may not be entirely true, as it reduces the diversity of individual trees, which is a unique advantage of random forests. But, we found that it slowed down the algorithm by increasing max_features. Therefore, we use grid searching to balance and choose the best max_features properly.

The n_estimators represent the number of subtrees before using the maximum number of votes or the mean to make predictions. A larger number of subtrees will give the model better performance, but at the same time take a longer time to fit.

Regarding min_sample_leaf, smaller leaves make it easier for the model to capture more features as well as more noise in the training data. In our project, we set the minimum number of leaf nodes to two (the default value) to 300 in order to search for the best model.

5.2. Task 2: Prediction

In task 2, we are required to predict the LoS of objects staying in the ICU based on the same features as in task 1.

5.2.1. Linear Regression

Linear regression (2) is one of the most easily interpretable models in regression tasks and always achieves acceptable results when the processed features are nicely selected. Nevertheless, it shares similar shortcomings as the Logistic regression models we introduced previously. We took this model in regression tasks and the Logistic regression model in classification tasks as baseline benchmark models. We adopted similar procedures in classification

tasks as introduced in 5.1.1 and we found that the features in Appendix B gave the best regression model.

$$\hat{y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_m X_m + \varepsilon \qquad (2)$$

5.2.2. Tree-based Models

Tree-based methods, as another non-parametric method we will use in our project, it is easy to visualize the decision trees. It selects features based on the largest gain in entropy at each step. Tree pruning will also be used to reduce the over-splitting of the training dataset. Besides, other tree construction methods will be introduced such as C4.5 and CART [12].

However, these simple building tree models are weak learners, as they may lead to undesirable predictions on their own. Furthermore, some ensemble learning methods including bagging, random forests, and boosting will be used to achieve better performances. Regarding bagging or bootstrap aggregation, it can reduce the variance of the statistical learning method but lose interpretability compared to decision trees. It takes many training sets called bootstrap samples from the population and builds separate models using each training set, and then averages the resulting predictions.

Random forests improve bagged trees by decorrelating the trees. Each time only a subset of predictors is chosen as split candidates, usually, the number of features in each split is equal to the square root of all the features. It is helpful if we have a large number of correlated predictors, which is a similar case in our project. Also, the overfitting problem is addressed to some extent in contrast to bagging.

In terms of boosting methods, trees are grown sequentially and strongly depend on those that have already been grown. Hence, the baseline model, the decision tree, in turn, can learn from mistakes or residuals in boosting. In our project, we used eXtreme Gradient Boosting (XGBoost) as our main boosting model. XGBoost uses both first- and second-order partial derivatives, with the second-order derivative facilitating faster and more accurate gradient descent. The use of Taylor expansions to obtain the second-order derivative form of the independent variable allows the leaf-splitting optimization to be performed solely on the value of the input data without selecting a specific form of the loss function, essentially separating the selection of the loss function from the optimization/parameter selection of the model algorithm. This decoupling increases the applicability of XGBoost, allowing it to select loss functions on demand, either for classification or regression.

5.3. Intended Advanced Methods

One of the largest challenges we encountered was feature engineering. With a total of 7488 features in the dataset, which makes all the classification and prediction models less robust and leads to bad results. Deep learning has achieved desirable results in selecting non-linear features automatically via the deep neural network. In this project, we will use an autoencoder to encode features and expect it to produce low-dimensional non-linear features for classification and prediction [2].

We have tried to tackle both tasks with simple neural network architectures, the multilayer perceptrons (MLP) with different numbers of hidden layers and neurons in each layer. Direct input of 7488 features and some feature sets with reduced dimensions have been trained via MLPs.

Regarding mortality prediction, when we used the raw 7488 features, the model quickly encountered overfitting even with one hidden layer and a dropout ratio of 0.6. Thus, we tried to fit the MLP with some feature sets with only several hundred features. We used the features stated in Appendix B together with some aggregations of means, standard deviations, ranges, and maximum values over the first 24 hours in the ICU. The hidden layers contain neurons of 128, 128, 32, and 8 and the dropout ratio of 0.5, 0.5, 0.3, and 0.1 respectively. Batch normalization is added in each hidden layer. A class weight of 1 to 2.3 is also acted on label 0 and label 1 respectively to encourage the model to learn more features for the minority class. This became the best MLP model with a training AUROC of 0.9204 and a validation AUROC of 0.8936. However, there are still some gaps between MLP and trees.

Turning to LoS prediction, we also used the same neural network architecture and best feature sets used in the classification tasks to directly train the regression task. However, it turned out that the best model did not perform very well, with a mean RMSE of about 1.78 on validation datasets. By trying different feature sets and different architectures, it finally achieved a mean validation RMSE of 1.75, which was not as good as the results we obtained by tree-based methods. This occurs probably because our data size is rather small compared to the features. And if we decreased the number of features, information would be less, and therefore performances would be worse. For these two particular tasks, we found that the non-parametric models like trees did outperform the other parametric models.

6. Results and Analysis

6.1. Task 1: Classification

In the logistic regression model using the original data, although the training accuracy (0.93) and the validation accuracy (0.94) seem to be high, but most data are predicted to be in the majority class. This results in a high false negative rate and an extremely low accuracy on the test dataset where the two classes are even. The model achieves a training AUC of 0.8685 and a validation AUC of 0.8337, which

is also relatively low.

We then tried to treat the dataset with different dimension reduction techniques before building logistic regression models. However, all of our dimension reduction methods failed to improve the AUC of our model or even degraded the performance of the model. The detailed ROC curves of the original logistic regression model and the logistic regression models with dimension reduction can be found in Figure 4.

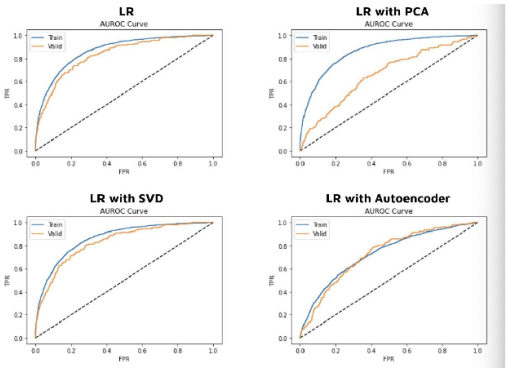

Figure 4. Comparison of LR with dimension reduction techniques

We use PCA as an example. It uses the explained variances as the criteria to get the principal components. The potential reason may be that in classification or regression tasks, the linear combinations of the features with large variances explained may not have too much prediction power in the prediction [9]. In other words, probably some ranges of medical indicators are causes for the mortality, and linear combinations of features simply eliminate such powerful predictors and hence resulted in undesirable results.

While undersampling and oversampling do not achieve satisfactory outcomes, the logistic regression model is improved when using data resampled by SMOTE. The AUC increases to 0.9081 for the training dataset and 0.8506 for the validation dataset. The training and the validation accuracy decrease to 0.8345 and 0.7916, respectively, possibly because more data are classified to class 1 and the false positive rate increases accordingly. The ROC curves of the original logistic regression model and the logistic regression models with data resampling can be found in Figure 5.

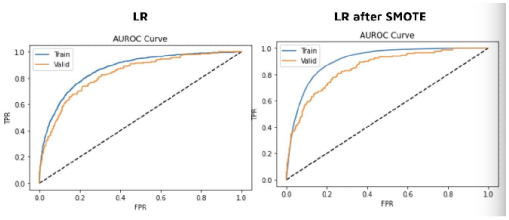

Figure 5. Comparison of LR with and without SMOTE

The LDA model achieves a training AUC of 0.9693 and a validation AUC of 0.8266. The result shows that the model has overfitting problems. The ROC curve is shown below in Figure 6.

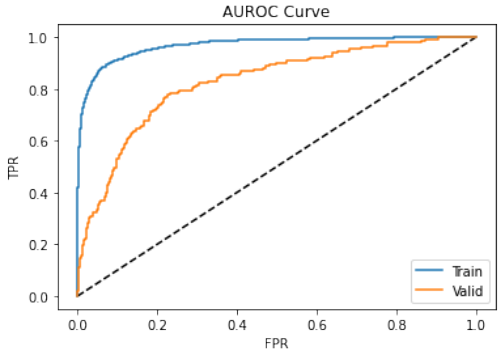

Figure 6. ROC curve of the LDA model

KNN model performs poorly and also encounters obvious overfitting problems in this task. When k is set to 3, the training AUC is 0.9877, and the validation AUC is 0.5487. Both training and validation AUC is even lower when k increases. The ROC curves of the KNN model using k=3 and k=30 can be found in Figure 7.

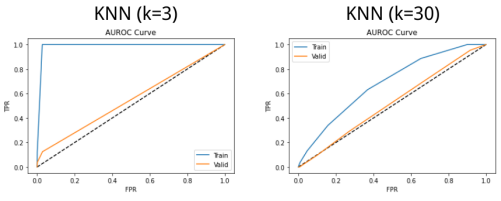

Figure 7. ROC curve of KNN models with different k values

Our best RF model is achieved with parameters n_estimators = 4000, min_samples_split = 200, max_samples = 4000, ccp_alpha = 0 and class_weight = "balanced_subsample". The accuracy AUC and validation AUC are 0.9076 and 0.8665, respectively. The ROC curve is shown in Figure 8.

XGBoost proves to be our best model for task 1. The parameters after fine-tuning are n_estimators=200, max_depth = 3, subsample = 0.95, colsample_bytree = 0.65, min_child_weight = 30, reg_lambda = 8 and gamma = 12. The accuracy AUC and validation AUC are 0.9395 and 0.9093, respectively. The ROC curve is shown in Figure 9.

6.2. Task 2: Prediction

The list of features used in the linear regression model can be found in Appendix B. We then filtered out 12 most important features using the feature_importances_ function

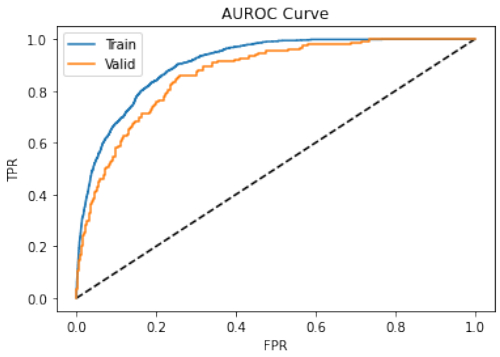

Figure 8. ROC curve of the best RF model

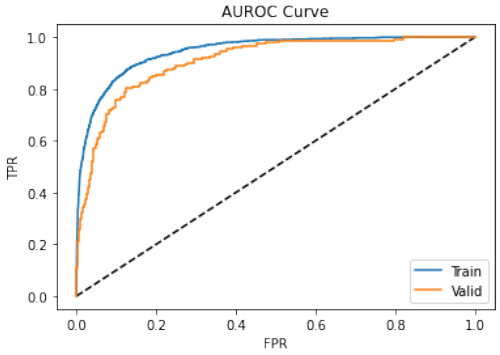

Figure 9. ROC curve of the best XGBoost model

with a threshold of 0.015. The linear regression is performed on the 12 variables together with their degree 2 polynomials. The RMSE of the LR model on the validation dataset is 1.7978.

The features for the tree-based models are selected according to the feature_importances_ function and filtered based on a certain threshold. The threshold for the RF model is 0.00019, and the parameters for RandomForestRegressor are n_estimators = 300 and max_depth = 17. The threshold for the XGBoost model is 0.0002, and the parameters for XGRegressor are n_estimators = 150, max_depth = 5 and colsample_bytree = 0.9. The RMSE of the RF model and XGBoost on the validation dataset is 1.7449 and 1.7359, respectively.

The final model for this task combines all three models we have tried, namely, linear regression, RF, and XGBoost. The weight for each model is found using linear regression on the predicted y in the validation dataset. The resulting weights are 0.1609359, 0.40743842, and 0.57090124 for each of linear regression, RF, and XGBoost model. By combining the three models, the RMSE of the final model is 1.6982, which is lower than the RMSE of any of the individual models, as shown in Figure 10.

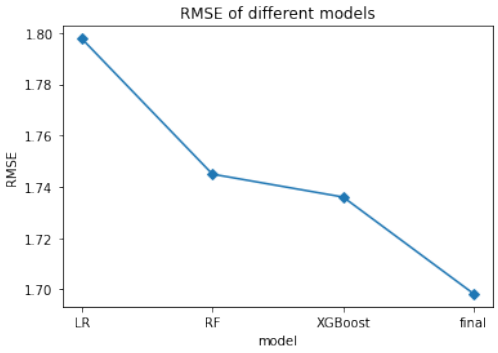

Figure 10. RMSE of different models

7. Limitation and Future work

Although the final models achieved excellent performance in both tasks, there remain some notable limitations, which require future work to resolve and improve. Despite that most features that we selected by consulting literature and adopting feature importance functions turned out empirically significant for the tasks, the theoretical interpretation of the feature selection process needs more research and exploration. In terms of the employed models, we encountered difficulties in taking advantage of the complexity of Multi-layer Perceptron (MLP), which tended to get overfitted while training. Even with means of modification such as regularization, use of dropout layers, and early stopping, it still under-performed the tree algorithms like random forest and XG-boosting. Therefore, further modification in network structure and fine-tuning of hyper-parameters will be conducted. Additionally, given the essence of two tasks, the idea of "multi-tasking" was naturally introduced. However, it has not been fully utilized in our project, and more relevant work will be done in the future.

8. Conclusion

In this group project, we experimented with several models and tried various means of improvement to tackle Task 1 and Task 2. We started with several baseline models, such as logistic regression, linear discriminant analysis, multilayer perception, etc., for Task 1 and multiple tree algorithms, etc. for Task 2. In the data pre-processing part, we referred to some literature, used the feature importance function, and tried unsupervised means to reduce the data dimension. While implementing models, we noticed that the dataset with imbalanced class labels might affect the performance of the algorithm; thus we turned to different sampling methods to diminish the bias. In order to fine-tune the hyper-parameters, the grid search method was adopted. With multiple experiments, it turned out XG-boosting outperformed all the other models we attempted both on Task

1 and Task 2. After analyzing and comparing the drawbacks across multiple models, we attempted to seek a way that mitigated the disadvantages of different models. Therefore, we took the linear regression of three models (namely linear regression, random forest, and XG-boosting), which performed significantly better than before, and that was our final model. In the future, more comprehensive exploratory data analysis will be conducted, and we will modify the architecture of MLP to manage its complexity better. We will also utilize multi-task learning to increase data efficiency and reduce over-fitting.

References

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010. 5

[2] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, page 899–907, Red Hook, NY, USA, 2013. Curran Associates Inc. 6

[3] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. Brits: Bidirectional recurrent imputation for time series, 2018. 1

[4] Peter Gnip, Liberios Vokorokos, and Peter Drotár. Selective oversampling approach for strongly imbalanced data. *PeerJ Comput. Sci.*, 7(e604):e604, June 2021. 3

[5] Tom Howley, Michael G Madden, Marie-Louise O'Connell, and Alan G Ryder. The effect of principal component analysis on machine learning accuracy with high-dimensional spectral data. *Knowl. Based Syst.*, 19(5):363–370, Sept. 2006. 2

[6] Alan Julian Izenman. Linear discriminant analysis. In *Springer Texts in Statistics*, Springer texts in statistics, pages 237–280. Springer New York, New York, NY, 2013. 5

[7] F. Li, and Y. Yang. A loss function analysis for classification methods in text categorization. *ICML*, 2003. 4

[8] Iftikhar Haider Naqvi, Khalid Mahmood, Syed Ziaullaha, Syed Mohammad Kashif, and Asim Sharif. Better prognostic marker in ICU - APACHE II, SOFA or SAP II! *Pak. J. Med. Sci. Q.*, 32(5):1146–1151, Sept. 2016. 1, 2

[9] M Pechenizkiy, A Tsymbal, and S Puuronen. PCA-based feature transformation for classification: issues in medical diagnostics. In *Proceedings. 17th IEEE Symposium on Computer-Based Medical Systems*. IEEE Comput. Soc, 2004. 2, 7

[10] Sanjay Purushotham, Chuizheng Meng, Zhengping Che, and Yan Liu. Benchmark of deep learning models on large healthcare mimic datasets, 2017. 1, 2

[11] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 4

[12] Steven L Salzberg. C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Mach. Learn.*, 16(3):235–240, Sept. 1994. 6

[13] Nicholas G. Polson, James G. Scott, and Brandon T. Willard. Proximal algorithms in statistics and machine learning. *Statistical Science*, 30:559–581, 2015. 4

[14] Shirly Wang, Matthew B. A. McDermott, Geeticka Chauhan, Marzyeh Ghassemi, Michael C. Hughes, and Tristan Naumann. MIMIC-extract. In *Proceedings of the ACM Conference on Health, Inference, and Learning*. ACM, apr 2020. 1

[15] Bechien U Wu, Richard S Johannes, Xiaowu Sun, Darwin L Conwell, and Peter A Banks. Early changes in blood urea nitrogen predict mortality in acute pancreatitis. *Gastroenterology*, 137(1):129–135, July 2009. 3

Appendices

Appendix A. FS-1

['temperature', 'blood urea nitrogen', 'glucose', 'heart rate', 'bicarbonate', 'systolic blood pressure', 'pulmonary artery pressure mean', 'glascow coma scale motor response', 'creatinine', 'glascow coma scale total', 'mean blood pressure', 'glascow coma scale eye opening', 'diastolic blood pressure', 'fraction inspired oxygen', 'white blood cell count', 'respiratory rate', 'ph urine', 'hematocrit', 'sodium', 'glascow coma scale verbal response', 'oxygen saturation', 'bilirubin', 'height', 'weight', 'potassium', 'ph', 'total protein urine']

Appendix B. List of Features from Literature

['glucose', 'creatinine', 'blood urea nitrogen', 'oxygen saturation', 'platelets', 'bicarbonate', 'diastolic blood pressure', 'heart rate', 'systolic blood pressure', 'mean blood pressure', 'respiratory rate', 'mean corpuscular hemoglobin concentration', 'mean corpuscular volume', 'anion gap', 'prothrombin time pt', 'partial thromboplastin time', 'phosphate', 'phosphorous', 'co2 (etco2, pco2, etc.)', 'partial pressure of carbon dioxide', 'lactate', 'glascow coma scale total', 'co2', 'neutrophils', 'lymphocytes', 'monocytes', 'positive end-expiratory pressure set', 'alanine aminotransferase', 'asparate aminotransferase', 'lactic acid', 'tidal volume set', 'basophils', 'albumin', 'tidal volume spontaneous', 'central venous pressure', 'fraction inspired oxygen set', 'lactate dehydrogenase', 'fibrinogen', 'fraction inspired oxygen', 'cardiac index', 'systemic vascular resistance', 'white blood cell count urine']