MLND Capstone Project: Predicting In-Hospital Mortality of ICU Patients

Ka Hung Wong

October 12, 2017

1 Introduction

1.1 Project Overview

Advanced and accurate warning of mortality risk could improve the quality and performance in the Intensive Care Unit (ICU) because it can achieve better clinical decision-makings and reduce the wait times for medical interventions [GIJ11, JQL+16]. Acuity scores which are calculated based on patient's features are currently applied in ICU to characterize the severity of a patient's illness. Unfortunately, these models have only modest specificity and sensitivity when they are implemented to predict the patient mortality [JQL+16]. This motivated PhysioNet to launch challenge in developing predictive model on in-hospital mortality in 2012 (PhysioNet 2012). The data used for the challenge is consisted of 5 general descriptors and 36 time series of vital signs and laboratory results from the first 48 hours of the first available ICU stay of patients from MIMIC-II dataset. Since it is difficult to have all measurements, it would be better to achieve prediction by a small subset of data. This proposal is inspired by the shortage of full dataset and aims to develop a prediction model based on a small subset of ICU data.

1.2 Problem Statement

The aim is to predict the in-hospital death (which is defined as positive when the length of stay in the hospital is longer than the survival time) based on a small subset of ICU data:

- the first 24 hours of the first available ICU stay
- 15 measurements

The steps required to achieve the goal are:

1. download the MIMIC-III data
2. build a local MIMIC-III PostgreSQL database
3. define the cohort
4. data pre-processing
5. train classifiers to predict the in-hospital death
6. compare with the benchmark

2 Metrics

In ICU, it is bad to have a lot of false negative since we do not want to miss any high risk patients. In this case, recall should be considered since higher recall indicates lower false negative rate.

However, a naive classifier that always predicts positive can achieve 100% recall, but it still a bad classifier since it does not classify at all. The shortcoming of recall can be addressed by considering precision, which is defined as the ratio of true positive to the predicted condition positive.

There are a lot of mapping from recall and precision to a single score, *e.g.*, simple average of them, can be used to represent the classifier performance. In our case, F1 score which is a harmonic mean of recall and precision is chosen because it is robust to unbalanced data. The mathematical expression of F1 score is:

$$\mathrm{F1} = \frac{2(\mathrm{recall}\mathrm{precision})}{\mathrm{recall} + \mathrm{precision}} \tag{1}$$

Matthews correlation coefficient (MCC) is another metric that is robust on unbalanced data, and it also consider the true negative. Therefore, it is chosen to compare the performance of the classifier in this project. The MCC is calculated:

$$\mathrm{MCC} = \frac{n_{TP} \times n_{TN} - n_{FP} \times n_{FN}}{\sqrt{(n_{TP} + n_{FP})(n_{TP} + n_{FN})(n_{TN} + n_{FP})(n_{TN} + n_{FN})}}, \tag{2}$$

where n_{TP} is the number of true positives, n_{TN} is the number of true negatives, n_{FP} is the number of false positives, and n_{FN} is the number of false negatives. Moreover, if any of the four sums in the denominator is zero, MCC is zero.

3 Analysis

3.1 Data Exploration and Exploratory Visualization

3.1.1 Gathering Data

The dataset used in the project is extracted from Medical Information Mart for Intensive Care (MIMIC-III) database [JPS+16]. MIMIC-III contains information related to patients admitted to critical care units at the Beth Israel Deaconess Medical Center in Boston, Massachusetts. The database contains data associated with 53,423 distinct hospital admissions for adult patients (aged $\geq$ 16 years) admitted to critical care units between 2001 and 2012. The database includes 26 tables containing admission information, patients' information such as physiological and laboratory measurements, and more.

MIMIC-III is installed in a local Postgres database on Windows. Tutorial on installing the database is available on MIMIC-III official website.

After installing the MIMIC-III database, we calculate the SAPS-II score using the data in the database. The code to create the materialized view for SAPS-II score from MIMIC-III is available in GitHub.

3.1.2 Data Extraction

Although there are 26 tables in the database, we only consider some of them. The tables related to the project include: *admissions*, *patients*, *icustays*, *chartevents*, *d_ items*, *outputevents*, *d_ labitems*, , *labevents*, and *SAPSII*.

First, we define the cohort based on three tables: *patients*, *admissions*, and *icustays*. The information contained in these tables are:

- *patients*: information about a patient that does not change
- *admissions*: information recorded on hospital admission
- *icustays*: information recorded on intensive care unit admission

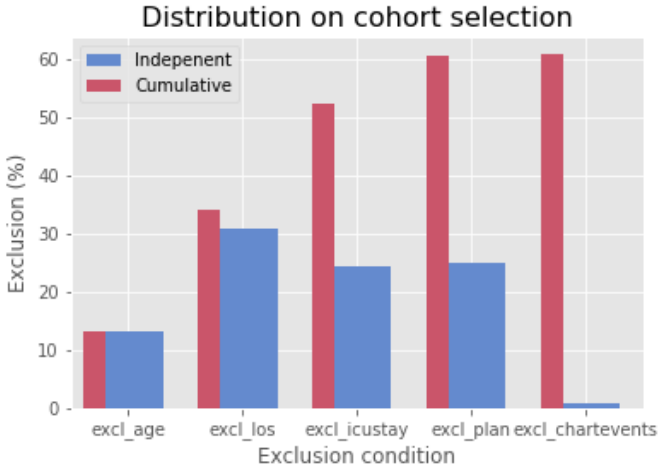

Figure 1: Exclusion distribution on cohort selection.

The constraints used in the cohort selection are:

- *excl_ age*: age $\geq$ 16 years
- *excl_ los*: length of ICU stay is longer than 1 day but less than 10 days to make sure the extracted data contains the first 24 hours of ICU stay
- *excl_ icustay*: only consider first ICU admission since patient can has several ICU admissions in one hospital admission
- *excl_ plan*: exclude the planned medical events
- *excl_ chartevents*: exclude patients with no physiological measurements

The exclusion distribution on cohort selection is shown in Fig. 1. Exclusion conditions on age, length of stay, first ICU stay, and expected medical event contributes most of the exclusions. Only a tiny portion of patients are excluded due to the missing medical measurements. After cohort selection, the number of observations is 24037, which is about 39% of the total observations in MIMIC-III.

Demographic of the cohort are then extracted, in which the in-hospital death, *ihd*, is calculated by

$$ihd = \begin{cases} 1, & \text{if discharge time} < \text{date of death} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The patients' demographic is saved into CSV format and is labelled as 'patient_details.csv'.

14 time series of physiological measurements from the first 24 hours of ICU stay of patients are extracted. The measurements includes bilirubin, blood urea nitrogen (BUN), fraction of inspired oxygen (FIO2), glasgow coma scale (GCS), bicarbonate (HCO3), partial pressure of oxygen (PO2), white blood cell (WBC), heart rate, potassium, respiratory rate (resp_rate), sodium, systolic blood pressure (sys_BP), temperature, and urine output volume (urine_out). These measurements are chosen since they are related with the calculation of SAPS-II score. Moreover, the SAPS-II score is extracted from the *SAPSII* table. Finally, these data is stored separately in 'chartdata.csv', 'labdata.csv', 'outputdata.csv', and 'sapsii_score.csv' files.

The scripts to extract these files from MIMIC-III database are available in 'data_extraction' Jupyter notebook.

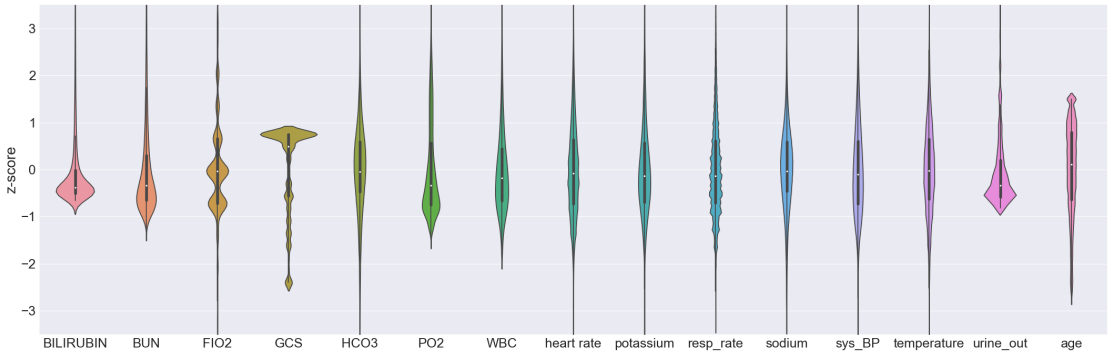

Figure 2: Distribution of z-score on the features.

3.1.3 Outliers

We first merge the data that extracted in the last section into two tables that describes the feature and target. 'chartdata.csv', 'labdata.csv', and 'outputdata.csv' are merged to represent the feature data. 'patient_details.csv' and 'sapsii_score.csv' are merged to represent the target data. The details of merging are shown in the Jupyter notebook, 'data_wrangling.ipynb'. The resulted feature data is described by a multi-index (patient's ID and time) multivariate time series with 15 various features, representing various medical measurements. The resulted target data is described by a 2D NX3 array, in which N is the number of observations. There are 3 variables in the target data, which are the in-hospital death, the SAPS-II score and its probability.

Data cleaning is mainly conducted for the feature data. We first define the outliers. The normal range of GCS is between 1 to 5, so the observations that are out of this range are considered to be outliers. For the 'age' variable, the range should be within 16 to 91.4 (the maximum age), and the values that are out of this range are considered to be outliers. For the rest of the variables, z-score is calculated and is used to define the outliers. The z-score is used compare an observation to a standard normal variable, and is calculated by:

$$z = \frac{x_{i,j}^k - E_j(x)}{\sigma_j(x)}. \tag{4}$$

Where $x_{i,j}^k$ represents the value of feature j in the observation at time i for patient with k ID. $E_j(x)$ is the expectation of x_j over all the time and patients, and $\sigma_j(x)$ is the standard derivation of x_j over all time and patients. In mathematical expression:

$$\begin{aligned} E_j(x) &= \frac{\sum_{i,k} x_{i,j}^k}{n} \\ \sigma_j(x) &= \sqrt{\frac{\sum_{i,k}(x_{i,j}^k - E_j(x))^2}{n-1}}. \end{aligned} \tag{5}$$

If the observation is not within 3.5 or -3.5 standard derivations from the mean, which suggests it is far away from the mean and can be considered as an outliers. All the outiers are considered as missing data, and further treatments are required. The z-score distribution after removing the outliers is shown in Fig. 2

3.1.4 Feature Correlation

Correlation matrix of the multivariate time series is calculated, and the matrix is visualized using a heatmap shown in Fig. 3. From the plot, there is no significant correlation between the features. Therefore, we should consider all the features in the further analysis.

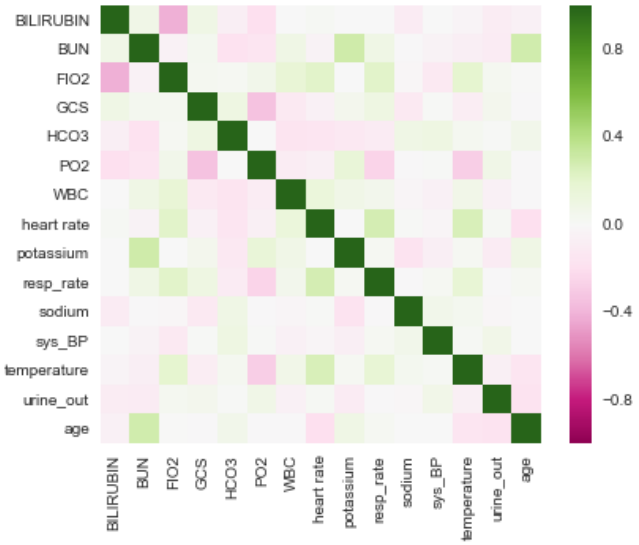

Figure 3: A plot of the correlation matrix of the multivariate time series.

Figure 4: The proportions of the missing data in the multivariate time series for the first 24 hours of ICU stay.

3.1.5 Missing Data

The extracted time series is unevenly spaced, and thus the time series is re-sampled with bi-hourly frequency. The resulted data contains a lot of missing data since the measurements do not cover the whole first 24 hours of the ICU stay. The ratio of missing data within the first 24 hours is summarized in Fig. 4. Ideally, we want a full multivariate time series that can capture the temporal information. Unfortunately, half of the features have more than 75% of missing data in this case. Only the urine output volume, respiratory rate, systolic blood pressure, heart rate, and age have more than 50% of the data.

3.1.6 Data Imputation and Transformation

The missing data is handled by imputation by the following steps:

- apply linear interpolation on the multivariate time series for each patient
- fill the missing data using the next valid observation for each patient

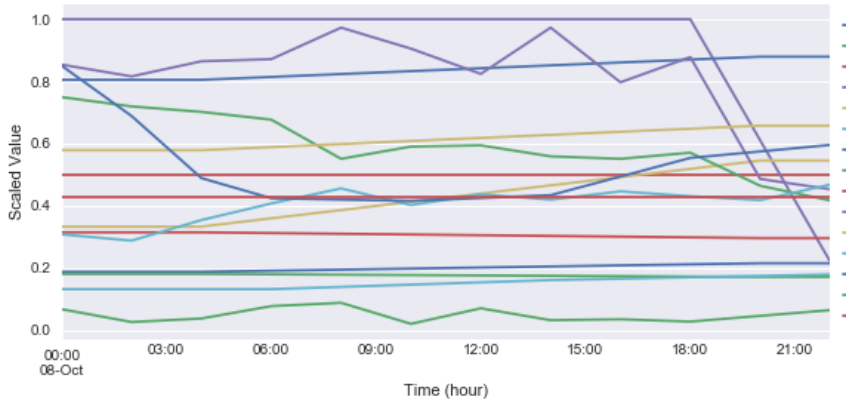

Figure 5: An example of imputed and transformed multivariate time series.

Item	Training time (s)	Predicting time (s)	F1 score	MCC score
DTW kNN	0.003	2244	0.2749	0.2612
Naive positive	NA	NA	0.22	0.0

Table 1: The result of DTW kNN and naive classifiers on a subset of training and testing datasets.

Some observations are still missing after these imputations, since there is entire missing data for some features for some patients. In this case, we fill the time series using the mean of all observations.

Finally, function 'MinMaxScalar' in Sklearn is applied to transform features by scaling each feature to the range of $[0, 1]$. One of the imputed and transformed multivariate time series is plotted as an example, which is shown in Fig. 5.

3.1.7 Data Splitting

The dataset is unbalanced, and it contains about 11.4% of positive samples. In order to preserve the percentage of samples for each class, stratified randomized split is applied to the dataset. The size of the test size is set to be 20% of the original dataset, which contains 4806 samples. On the other hand, the training dataset contains 19220 samples. Finally, the training and testing datasets are split into 'X_train', 'X_test', 'y_train', and 'y_test'. A sample of the pre-processed feature (X) and target (y) data is shown in Tables 3 & 4.

3.2 Algorithms and Techniques

Approaches to classify time series can be categorized into three types [XPK10]: (1) representation methods, which extract new features for representing the time series, (2) model generation, which assumes time series in a class is generated by an underlying model, and (3) similarity measures, which calculate the similarity between different time series.

One of the similarity measures is dynamical time wrapping (DTW), which calculates the best matches between two time series. Once the distance-like similarity between two time series is defined, classical distance based algorithm such as k-NN, SVM can be applied to classify the time series. DTW with kNN is one of the best solutions known for time series problems in a variety of domains [RK04]. Therefore, we first attempt to classify our medical multivariate time series using DTW with kNN classifier.

The DTW and kNN algorithms are implemented via Python in 'dtw.py' and 'KNN.py'. Since the time complexity of DTW and kNN are $O(n^2)$ [RK04] and $O(ndk)$, we first train the model with

a subset of training and testing data for better time management. 1922 (10%) and 480 (10%) training and testing observations are used. The result of DTW 3-NN classifier is summarized in Table 1. The k-NN classifier, the lazy learner, does not train a model, and its training time is negligible. However, it took about 2244 seconds to predict the label of the subset of testing data which suggests it is computational expensive. The performance of DTW k-NN prediction is not promising since the naive classifier that always predicts positive can achieve a comparable F1 score to it. Although using the full training dataset could improve the performance of DTW kNN, the computational cost is too expensive. Moreover, the missing rate of the multivariate time series is high as shown in Fig 4, which indicates some variables may contain less temporal information. The loss of temporal information is shown to have negative impact on DTW based classifier which is demonstrated in the literature [MBSRJ].

The second attempt in this project is to extract new features from the multivariate time series. This removes the time dependence of the data and reduces the training data from 3 dimensions to 2 dimensions such that the usual classification methods such as logistic regression are applicable. In this project, logistic regression, decision tree, gradient boost, and multilayer perceptron classifiers are used.

Logistic regression is a simple which is less prone to over-fitting. It has a fast training time and is robust to noisy data. Therefore, it is used as a baseline classifier. In logistic regression, we train a function of the form,

$$P(y = 1|x) = \sigma(\theta^T x) = \frac{1}{1 + \exp(-\theta^T x)}, \tag{6}$$

to minimize the cost function:

$$J(\theta) = -\sum_i \left(y_i \log\left(P(y = 1|x_i)\right) + (1 - y_i) \log\left(1 - P(y = 1|x_i)\right)\right), \tag{7}$$

for all samples, $x_i \in \mathbb{R}$. This returns θ, which is used to calculate the probability of being positive or negative based on x.

Decision tree is a logical based learner, which it recursively partitions the sample space such that the samples in the same class are in the same boundary. In Sklearn, the default learner is trained to reduce average Gini impurity to have information gain per iteration. The average Gini impurity is:

$$I_G(S, A) = \sum_i \frac{|S_i|}{|S|} I_G(S_i), \tag{8}$$

where S_i is the partition size, $I_G(S_i)$ is the Gini impurity of partition S_i:

$$I_G(S_i) = 1 - \sum_j p_j^2. \tag{9}$$

In the equation, p_j represents the proportion of examples in class j over S_i.

Gradient boost classifier is one of the ensemble methods that constructs and trains a set of learners. The updated learner set is:

$$F_{i,m+1}(x) = F_{i,m}(x) + \eta h_i(x), \tag{10}$$

where η is a learning rate, $h_i(x)$ is the new learner, $F_{i,m}$ is the model for class i and is used to calculate the probability of being in class i by:

$$P_i = \frac{\exp(F_i)}{\sum_i \exp(F_i)}. \tag{11}$$

The new learner can be calculated by minimizing the KL-divergence:

$$h_i(x) = y_i(x) - P_i(x) \tag{12}$$

where y_i is the true probability (the true labelling). Predictions made from these learners are thus weighted giving an overall prediction to the problem. Furthermore, boosting can control both bias and variance to enhance the predicting performance. However, it is computational expensive because multiple models are required. Since many learners are involved, it is difficult to interpret the prediction made.

Multilayer perception is different from logistic regression since there are one or more hidden layers between the input and the output layer. Each neutron, h_j, in hidden layers receive signal from the previous layer and fire signal to the next layer. In mathematical, we have:

$$h_j = f_h \left(w_{j,0} + \sum_i g_i(x) w_{j,i} \right), \tag{13}$$

where $g_i(x)$ is the output signal from the previous layer, and $w_{j,i}$ is the weight to get a linear combination of signals from the previous layer. The function f_h is the activation function returning signal based on the previous input signals. The weight is updated by back-propagation, in which the error gradient ($\frac{\partial E}{\partial w_{j,i}}$) with respect to $w_{j,i}$ is calculated. The hidden layers allows the MLP to learn complex non-linear model to provide good results. However, it may experience over-fitting and is quite sensitive to the initial weights and the scaling of the data.

3.3 Benchmark

Simplified acute physiology score (SAPS) II is used as benchmark model. SAPS II is one of the acuity scores widely applied in clinical studies [GLS93]. It is based on 12 physiological variables in the first 24 hours of ICU admission and admission health status of patients (SAPS score calculation). SAPS II was calibrated to allow conversion of the calculated score into the probability of mortality [GLS93]. In order to calculate the F1 and MCC scores of the benchmark, a threshold is required to convert the probability to the predicted label. In this project, we set the threshold to be 0.5, *i.e.,* if the probability based on SAPS-II is larger than 0.5, then in hospital death is predicted. In mathematical term:

$$l_{SAPS} = \begin{cases} 1, & \text{if } P_{SAPS} > 0.5 \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

The SAPS-II probability and prediction label are stored in 'y_test' and 'y_train'. The F1 and MCC scores of SAPS-II scoring system will compare with the F1 and MCC scores of the machine learning classifiers.

4 Methodology

4.1 Data Preprocessing

Re-sampling and imputation on the feature matrix, X, are done in the exploratory data analysis. Then, the feature matrix is used to train a DTW kNN classifier for the first attempt.

Further data preprocessing is required for the second attempt, in which conventional supervised algorithms can be used. For k-th patient, we have a 2D array of data, $X_k = (x^k_{i,j})_{n,m}$ with shape (n, m), where k is the k-th patient, n is the number of observations during the first 24 hours of ICU stay, and m is the number of measurements. We then calculate several statistical properties

based on the 2D array:

$$\begin{aligned}
\max_{k,j} &= \max_i(X_k) \\
\min_{k,j} &= \min_i(X_k) \\
\text{median}_{k,j} &= \text{median}_i(X_k) \\
\mu_{k,j} &= \sum_i x_{i,j}^k/12 \\
\sigma_{k,j} &= \sqrt{\left(x_{i,j}^k - \mu_{k,j}\right)^2/12}.
\end{aligned} \tag{15}$$

Then, we calculate the global Q1 and Q3 of the variable over all patient information:

$$\begin{aligned}
\text{Q1}_j &= \text{Q1}(x_{i,j}^k) \\
\text{Q3}_j &= \text{Q3}(x_{i,j}^k).
\end{aligned} \tag{16}$$

Then we count the number of data points that is above or below the Q3 or Q1.

$$\begin{aligned}
cQ1_{k,j} &= \sum_i (x_{i,j}^k < Q1_j) \\
cQ3_{k,j} &= \sum_i (x_{i,j}^k > Q3_j).
\end{aligned} \tag{17}$$

For each variable j, we have 7 new features extract from the time series. Therefore, the original data (with shape of (k,n,m)) is transformed to a 2D array with shape of (k,m) which can be applicable to conventional classifiers such as logistic regression.

4.2 Implementation

4.2.1 Metrics

'f1_score' and 'matthews_corrcoef' are imported from the 'sklearn.metrics' module to calculate the F1 score and the Matthews correlation coefficient.

4.2.2 algorithms

For the first attempt that uses DTW with kNN classifier. A custom 'dtw.py' and 'KNN.py' scripts are implemented. 'distance.euclidean' from scipy library is imported to use as the distance metric to calculate the similarity between two time series. Also, the 3 neighbors is chose in the k-NN classifier.

For the second attempt, 'LogisticRegression', 'DecisionTreeClassifier', 'GradientBoostingClassifier', and 'KerasClassifier' are imported from 'sklearn.linear_model', 'sklearn.tree', 'sklearn.ensemble', and 'keras.wrappers.scikit_learn' to build the corresponding learners.

First, we set up a pipeline to train all four classifiers to the training data to obtain a preliminary results. Default classifiers are used in 'LogisticRegression', 'DecisionTreeClassifier', 'GradientBoostingClassifier'. Random state is fixed to zero for reproducibility. The code to create these classifiers are:

```
clf_A = LogisticRegression(random_state = 0)
clf_B = DecisionTreeClassifier(random_state = 0)
clf_C = GradientBoostingClassifier(random_state=0)
```

The MLP architecture is shown here:

```
def keras_model():
    clf_MLP = Sequential()
    clf_MLP.add(Dense(128, activation='relu',
    input_shape= data_agg['X_train'].shape[1:]))
    clf_MLP.add(Dropout(0.1))
    clf_MLP.add(Dense(32, activation='relu'))
    clf_MLP.add(Dropout(0.1))
    clf_MLP.add(Dense(8, activation='relu'))
    clf_MLP.add(Dropout(0.1))
    clf_MLP.add(Dense(1, activation='sigmoid'))
    clf_MLP.compile(optimizer= 'adam', loss='binary_crossentropy')
    return clf_MLP
```

The number of neutrons is chosen to make the total number of parameters (17969 for this MLP) less than the training data which is 19220 to reduce the chance of over-fitting. The default settings of dense layer are:

```
default = {use_bias=True, kernel_initializer='glorot_uniform',
bias_initializer='zeros', kernel_regularizer=None, bias_regularizer=None,
activity_regularizer=None, kernel_constraint=None, bias_constraint=None}
```

5 Results

5.1 Model Evaluation and Validation

The training data contains 19220 samples, and the testing data contains 4806 samples. We train the learners by different sizes, 1%, 10%, and 100% of the training data. We calculate the scores using the full training and testing sets.

Default settings are applied for the Sklearn's learners. However, there is no default architecture for the multilayer perceptron. Therefore, we build a simple one by adding two small hidden layers between the input and output layers. The number of perceptron is (128, 32, 8, 2) with 14138 trainable parameters. 'relu' activation function is applied in the first three layers, while 'sigmoid' activation function is applied in the output layer. Several dropout layers are added to prevent overfitting.

5.2 Justification

Decision tree shows the poorest result as indicated in Fig. 6. The decision tree archives the highest F1 score and MCC in the training data. However, its performance in testing data is worse than the benchmark which strongly suggests the decision tree learner is over-fitted during the training process.

Logistic regression, gradient boost, and multilayer perceptron learners achieve better performances than the benchmark in testing data as shown in Fig. 6.

Although logistic regression is a simple classifier, it works better than the benchmark by achieving a higher Matthews correlation coefficient. Also, by comparing the scores between training and testing sets, it does not overfit the learner.

Both gradient boost and multilayer perceptron achieve good improvements and performs better the logistic regression classifier. The choice between gradient boost and multilayer preceptron depends on the number of samples in the training data. When the data is enough (100%), multilyer perceptron can perform better than gradient boost. However, it perform badly when the sample size is small (1% or 10%). Since we can assume there is more medical data in the future, multilayer preceptron could have a better position. One of the drawback of gradient boost and mulitlayer

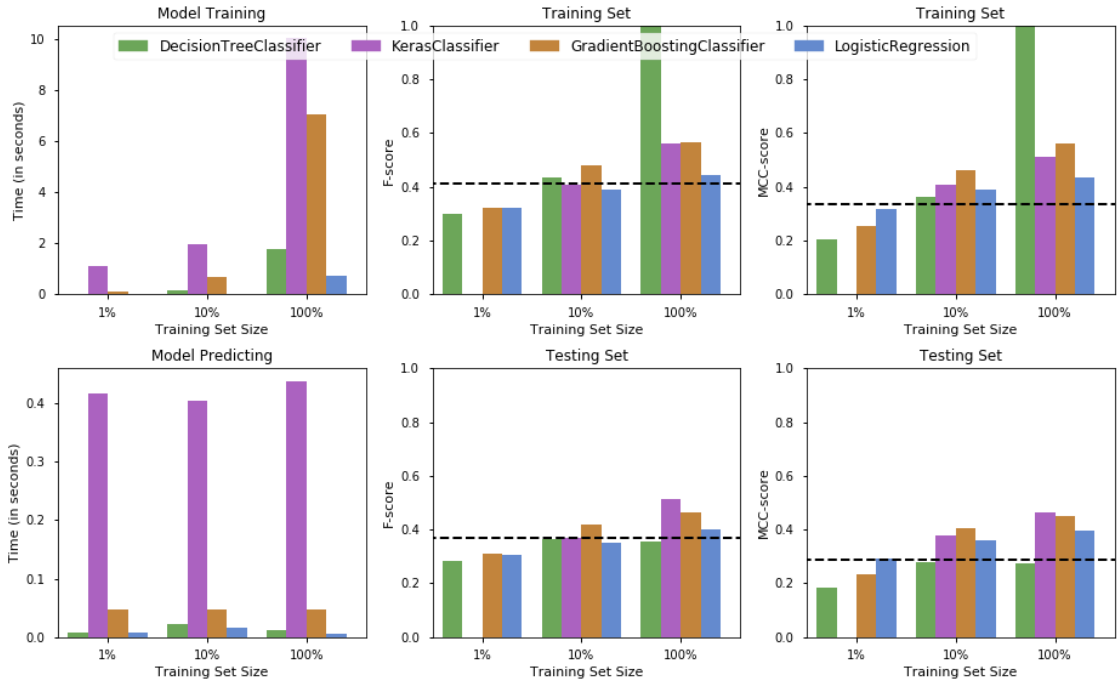

Figure 6: Performance metrics for logistic regression, decision tree, gradient boost, and multilayer perceptron classifiers. Black line represents the F1 score and MCC of the benchmark.

Classifier	F1	MCC
SAPS-II	0.371, 0.290	
Initial Gradient boost	0.465	0.452
Initial Multilayer perceptron	0.511	0.474
Tuned Gradient boost	0.484	0.448
Tuned Multilayer perceptron	0.485	0.447

Table 2: The performance of initial and tuned gradient boost and multilayer preceptron classifiers. The SAPS-II scores in F1 and MCC are shown for comparison.

perceptron are required a longer training time. However, the training time (< 20 s in our dataset) is still acceptable providing there is an improvement in prediction performance.

We further tune the hyper-parameters of the gradient boost and multilayer perceptron classifiers. For the gradient boost classifier, learning rate, number of estimators, maximum of the depth, minimum of samples slip, and minimum samples leaf are considered. First, we search over the learning rate, $[0.1, 0.2, 0.3]$, and number of estimators, $[60, 80, 100]$. The result suggests the best performance achieved when the learning rate is 0.2 and the number of estimator is 100, and we use these values for next tuning. The maximum depth and minimum samples split are searched in the following space $[3, 5, 7, 9, 11] \times [100, 300, 500, 700, 900]$. Also, we use the values in the best result for the next tuning in the minimum samples leaf with the sample space of $[1, 21, 41, 61]$. The final hyper-parameters used in the last implementation is:

```
tuned_GB_clf = GradientBoostingClassifier(random_state = 0, learning_rate=0.2,
n_estimators=100, max_depth=5, min_samples_split=100, min_samples_leaf=1)
```

By comparing with initial gradient boost classifier,

```
inital_GC_clf = GradientBoostingClassifier(random_state = 0, learning_rate=0.1,
n_estimators=100, max_depth=3, min_samples_split=2, min_samples_leaf=1)
```

the tuned classifier has a slower learning, a bigger maximum depth of the individual estimators, and

bigger minimum number of samples to split. With these tuned parameter, F1 score on predicting testing data slightly increases as shown in Table 2. However, there is a little bit drop in MCC score. It is not evident that the tuned model is better than the initial model the tuning is using 5-fold cross validation with 19220 of training data.

For the multilayer perceptron, we can see the over-fitting is not significant in the initial training (by comparing the training and testing scores in Fig. 6). Therefore, we keep the dropout rate and the number neutron in the layers. First, we focus on the batch size and the number of epochs. Then, we move on to the weight initialization.

The batch size and epochs are chosen to create a sample space of $[32, 256, 512] \times [10, 50, 100]$. The 5-fold cross validation results suggests the best performance achieved when the batch size and epochs are 512 and 50, and this is possibly due to the reduction of noise in training signal when the learner updates it weight. Then, we tune the MLP with initial weight by considering 8 different weight initializers available in Keras model: uniform, lecun_uniform, normal, zero, glorot_normal, glorot_uniform, he_normal, he_uniform. Weight initialization is important to MLP because the MLP learner could find local optimum instead of global optimum. After calculating the 5-fold cross validation, the results suggests the default initializer, glorot_uniform, achieves the best performance. The final tuned hyper-parameters are the MLP are:

```
tuned_MLP_clf = Sequential()
tuned_MLP_clf.add(Dense(128, kernel_initializer='glorot_uniform',
activation='relu', input_shape= data_agg['X_train'].shape[1:]))

tuned_MLP_clf.add(Dropout(0.1))

tuned_MLP_clf.add(Dense(32, kernel_initializer='glorot_uniform',
activation='relu'))

tuned_MLP_clf.add(Dropout(0.1))

tuned_MLP_clf.add(Dense(8, kernel_initializer='glorot_uniform',
activation='relu'))

tuned_MLP_clf.add(Dropout(0.1))

tuned_MLP_clf.add(Dense(1, kernel_initializer='glorot_uniform',
activation='sigmoid'))

tuned_MLP_clf.compile(optimizer='adam', loss='binary_crossentropy')

tuned_MLP_clf.fit(data_agg['X_train'].values,
data['y_train']['in_hospital_death'], batch_size=512, epochs=50,
verbose=1, shuffle=True)
```

Finally, the F1 and MCC scores of predicting the testing dataset is generated using the tuned MLP model as shown in Table 2. Surprisingly, the tuned performance is worse than the initial performance. Since the initial MLP batch size, epochs, and weight initializer are 32, 10, and 'glorot_uniform'. We look at the 5-fold cross validation result on the training data with the same hyper-parameters, and we found that the mean F1 score is 0.481 with the standard derivation of 0.039. This may suggest the F1 score, which is 0.511, on predicting the testing dataset using initial MLP model is not that significantly different from the mean F1 score obtained in 5-fold cross validation on the training dataset. Therefore, it is not evident enough that the initial MLP is better than the tuned MLP.

Since the tuned models are evaluated via the 5-fold cross validation on the training dataset. The means obtained is more stable than the results obtained just using the testing dataset. Therefore, the tuned model could possibly more robust to the problem.

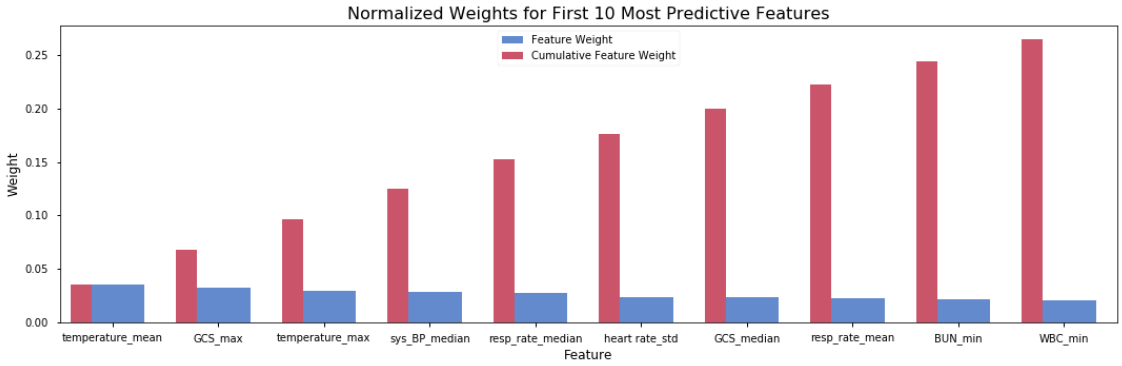

Figure 7: The first 10 most important features related with in-hospital death.

After tuning the hyper-parameter, gradient boost and MLP classifiers can achieve a F1 score that closes to 0.5 as shown in Table 2, which about 0.1 higher than the benchmark using SAPS-II score. This also shows the power of machine learning since we achieve a better prediction using the almost same data used to calculate the SAPS-II score, which a complex scoring system developed by medical experts.

6 Conclusion

The aim of the project is to achieve a better predicted power on in-hospital death in ICU based on the same dataset with the medical expert system, SAPS-II scoring. Since the tuned MLP and gradient boost classifiers obtain a significantly higher F1 and MCC scores than the scores obtained by using SPAS-II system, it suggests these learners have the potentials to use in ICU to achieve better clinical decision-makings and reduce the wait times for medical interventions.

6.1 Improvement

Although there is an improvement in prediction using gradient boost and multilayer preceptron classifiers, there is still plenty of room for improvement based on the F1 and MCC scores achieved in both classifiers.

Up to the last section, we focus on the algorithm. In this section, we do some analysis on the data itself. First, we determine the variables in the data provides the most predictive power. Using gradient boost classifier, we first fit the classifier and then return the feature_importance_ attribute. Top 10 most important variables are shown in Fig. 1. We can see that the cumulative weight of the top 10 most important variables is still less than 30%. This suggests the mutual information between each variable and in hospital death is small. The variety of the data (15 time series) that was extracted from MIMIC-III database may be not enough to achieve a high completeness. Therefore, one of the approaches to improve the predictive power is using the full MIMIC-III. After achieving a good prediction, then we can reduce the dimensions by various statistical methods such as PCA.

6.2 Reflection

MIMIC-III is a rich database, about 50% of the time spent in the project is on data extraction and data preprocessing. there are several difficulties during this process:

- starting with minimal knowledge on medical science
- learning how to write complex SQL scripts

- deciding the final format of the extracted data
- handling the missing data

Classification of multivariate time series is not straightforward, and I learned a lot during solving the problem. At the beginning, the literature review suggests me to apply DTW with kNN on the problem. However, during the implementation, I found it is slow and the result is not that promising. So I need to find another approaches to solve the problem. At that time, I realized the usual project management can be very important to data science project. For example, if there is a more detailed planning (*e.g.*, with several backup plans) at the beginning, I believe I can achieve better execution efficiency.

References

[GIJ11] Siontis GM, Tzoulaki I, and Ioannidis JA. Predicting death: An empirical evaluation of predictive tools for mortality. *Archives of Internal Medicine*, 171(19):1721–1726, 2011.

[GLS93] Jean-Roger Le Gall, Stanley Lemeshow, and Fabienne Saulnier. A new simplified acute physiology score (saps ii) based on a european/north american multicenter study. *JAMA*, 270(24):2957–2963, 1993.

[JPS+16] Alistair E.W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3:160035, 2016.

[JQL+16] Calvert Jacob, Mao Qingqing, Hoffman Jana L, Jay Melissa, Desautels Thomas, Mohamadlou Hamid, Chettipally Uli, and Das Ritankar. Using electronic health record collected clinical variables to predict medical intensive care unit mortality. *Annals of Medicine and Surgery*, 11:52–57, 2016.

[MBSRJ] Karl Oyvind Mikalsen, Filippo Maria Bianchi, Cristina Soguero-Ruiz, and Robert Jenssen. Time series cluster kernel for learning similarities between multivariate time series with missing data. *arXiv*.

[RK04] Chotirat Ann Ratanamahatana and Eamonn Keogh. Everything you know about dynamic time warping is wrong. 2004.

[XPK10] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. A brief survey on sequence classification. *SIGKDD Explor. Newsl.*, 12(1):40–48, November 2010.

A Sampling of feature data

Table 3: The first 36 rows of data on the pre-processed testing dataset representating the first 24 hours medical measurements of 3 patients. Features 'potassium', 'resp_rate', 'sodium', 'sys_BP', 'temperature', 'urine_out', and 'age' have been omitted for better printout.

id	time	BILIRUBIN	BUN	FIO2	GCS	HCO3	PO2	WBC	heart rate
19220	0	0.082	0.105	0.499	1.000	0.545	0.220	0.251	0.638
19220	1	0.082	0.105	0.499	1.000	0.545	0.220	0.251	0.669
19220	2	0.082	0.105	0.499	1.000	0.545	0.220	0.251	0.606
19220	3	0.082	0.105	0.499	1.000	0.545	0.220	0.251	0.614
19220	4	0.082	0.105	0.499	1.000	0.545	0.220	0.251	0.512
19220	5	0.082	0.105	0.499	1.000	0.545	0.220	0.251	0.472
19220	6	0.082	0.105	0.499	1.000	0.545	0.220	0.251	0.496
19220	7	0.082	0.105	0.499	1.000	0.545	0.220	0.251	0.528
19220	8	0.082	0.105	0.499	1.000	0.545	0.220	0.251	0.630
19220	9	0.082	0.105	0.499	1.000	0.545	0.220	0.251	0.567
19220	10	0.082	0.105	0.499	1.000	0.545	0.220	0.251	0.457
19220	11	0.082	0.105	0.499	1.000	0.545	0.220	0.251	0.457
19221	0	0.082	0.124	0.499	1.000	0.394	0.343	0.173	0.530
19221	1	0.082	0.124	0.499	0.750	0.394	0.343	0.173	0.505
19221	2	0.082	0.124	0.499	0.500	0.394	0.429	0.173	0.480
19221	3	0.082	0.124	0.499	0.833	0.394	0.429	0.173	0.496
19221	4	0.082	0.124	0.499	1.000	0.394	0.429	0.173	0.453
19221	5	0.082	0.124	0.499	1.000	0.394	0.429	0.173	0.508
19221	6	0.082	0.124	0.499	1.000	0.394	0.429	0.173	0.437
19221	7	0.082	0.124	0.499	1.000	0.394	0.429	0.173	0.421
19221	8	0.082	0.124	0.499	0.918	0.394	0.429	0.173	0.406
19221	9	0.082	0.124	0.499	0.918	0.394	0.429	0.173	0.476
19221	10	0.082	0.124	0.499	0.918	0.394	0.429	0.173	0.465
19221	11	0.082	0.124	0.499	0.833	0.394	0.429	0.173	0.457
19222	0	0.133	0.457	0.499	1.000	0.636	0.220	0.159	0.402
19222	1	0.133	0.457	0.499	1.000	0.636	0.220	0.159	0.402
19222	2	0.133	0.457	0.499	1.000	0.636	0.220	0.159	0.409
19222	3	0.133	0.457	0.499	1.000	0.636	0.220	0.159	0.386
19222	4	0.133	0.457	0.499	1.000	0.636	0.220	0.159	0.382
19222	5	0.133	0.457	0.499	1.000	0.636	0.220	0.159	0.390
19222	6	0.133	0.457	0.499	1.000	0.636	0.220	0.159	0.386
19222	7	0.133	0.457	0.499	1.000	0.636	0.220	0.159	0.417
19222	8	0.133	0.457	0.499	1.000	0.636	0.220	0.159	0.394
19222	9	0.133	0.457	0.499	0.791	0.636	0.220	0.159	0.409
19222	10	0.133	0.457	0.499	0.583	0.636	0.220	0.159	0.412
19222	11	0.133	0.457	0.499	0.500	0.636	0.220	0.159	0.396

B Sampling of target data

Table 4: The first 24 rows of data on the pre-processed testing target dataset representating 24 patients' information.

id	in_hospital_death	sapsii_prob	sapsii_prediction
0	0	0.096697836	0
1	0	0.391925539	0
2	0	0.140051063	0
3	0	0.096697836	0
4	0	0.152870496	0
5	0	0.140051063	0
6	0	0.041753508	0
7	0	0.033061822	0
8	0	0.041753508	0
9	0	0.037204666	0
10	0	0.071716185	0
11	0	0.152870496	0
12	1	0.941142064	1
13	0	0.266086521	0
14	0	0.087706213	0
15	0	0.07939038	0
16	1	0.285486456	0
17	0	0.229591669	0
18	0	0.106398223	0
19	0	0.064648836	0
20	0	0.166522917	0
21	0	0.24744285	0
22	0	0.181019762	0
23	0	0.019975877	0
24	0	0.029295167	0