Machine Learning Engineer Nanodegree

Capstone Project report

Wenbo Wang

April 1st, 2018

I. Definition

Project Overview

According to centers for medicare and medicaid serves (CMS), US national healthcare expenditure (NHE) grew by 4.3% to $3.3 trillion in 2016 and accounted for 7.9% of Gross Domestic Product (GDP). The NHE projection stated that 5.5% per year growth is expected, reaching $ 5.7 trillion by 2026 [1]. Therefore, health care expenditure continued to rise without us knowing if the growing NHE indeed resulted in better or more timely health care services that lead to lower mortality rates and improved prognosis for the patients. At the same time, it was estimated that $ 750 Billion was wasted each year in health care system due to many reasons [2].Some of the major reasons that have been identified for US health care waste are over-testing, over-priced medical tests and procedures, inefficient delivered services,and missed prevention opportunities, etc.Thus,the many causes for health care waste, which could be addressed through policy making or technology innovation, could potentially be remedied upon further investigation.Many lives could be saved without further investing if resources could be optimally utilized in hospitals.

With an aging population in the US and many developed nations, there is an urgent call for health care system to provide medical service to patients are in needs, targeting treatments towards individuals rather than the general population. However, the health care system is a highly complicated network of standards and clinical guidelines that aimed to provide effective treatment in the aggregate for population in general.Despite the widely different medical conditions upon admission, patients are routed to clinical procedures according to established practices instead. Thus, this potentially lead to patients with less severe conditions taking up limited ICU resources while patients with acute illness was delayed in receiving proper treatment.

The question here remained is how we are able to make better clinical decisions based on what is available. Implementation of electronic health care record (EHS) system has resulted in aggregation of gigantic amount of data coming from various sources, e.g., monitoring instrument outputs, clinical records, and patient history data.etc. Therefore, the 'big data' may offer a solution to the existing health care problem at hand.One of the widely cited and utilized public medical database is the medical information mart for Intensive Care (MIMIC-III)[3]. This large, single-center database provided by laboratory for computational physiology at Massachusetts Institute of Technology composed of various types of medical data, such as laboratory measurements, vita signs, observations, procedure codes, survival data and more.This dataset is collected over 10 years between 2001 and 2012 with over 53,000 distinct hospital admissions.

While being entertained by the idea that aggregated data may provide an answer, at the same time, the highly heterogeneous medical data presented great challenges in its interpretation. Decisions of inclusion and exclusion of particular features, handling missing values and extreme measurement data points requires in-depth field expert knowledge in medicine as well as vast amount of expertise in data science. Therefore, the prospective of offering more precise clinical decision support based on analysis of large-scale medical database has yet to weigh on selection of proper data analysis methods, particularly machine learning algorithms,along with the available computing facilities. Eventually, as both the data availability and domain expertise in handling such enormous amount of data grow, the trend to use such big data approach to come up with timely predictions may either complement or even replace existing clinical guidelines will become a reality that means substantial health care savings and greatly decreased mortality rate.

Problem Statement

The timeliness and appropriateness of treatment at the right hospital facility plays significant role in determining the prognosis of admitted patients. Thus accurate prediction of patient sickness severity both benefit patient and make efficient use of hospital resources. One of the primary outcome from hospital admission is in-hospital death.Many severity scoring systems such as Acute Physiology and Chronic Health Evaluation (APACHE) II scores, Simplified Acute Physiology Score (SAPS) II scores, etc have been proposed[4,5]. Those scores were usually calculated based on a set of hand-picked predictor variables, with some knowledge from medical practitioners or simply using an ad hoc approach, to calculate patient mortality probability [6]. Though proven to be an effective measure to gauge patient health condition at admission, these scoring system definitely leave plenty of room for improvement. Statistical learning takes in significantly larger dataset and look at more features simultaneously, trying to figure out the inner relationship between predictor variables and optimal structures to come up with robust prediction models for future unseen data.Thus, I am trying to come up with mortality rate prediction models that generalize better comparing to existing severity score SAPS systems.

In this capstone project, I explored MIMIC-III database by querying and aggregating various data sources, where severity scoring predictors construed a subset, that are believed to be most pertinent to predicting patient in-hospital outcomes.Setting up and querying a relational database at the scale of MIMIC-III is a challenge itself, but this is very helpful because of the widespread use of such data storage models in health care system, making it a valuable skill for data science practitioners.As complicated as the medical problems,electronic medical information is somewhat contains large volume of information that could contains errors, missing values,or artificially created data entries for privacy purposes. Thus, cleansing and extracting data require both some field knowledge and judgement calls as a data scientist. Because of the inherent data characteristics, machine learning approaches should be somewhat tolerant of outlying measurements and providing models that is understandable to medical practitioners. Then, I train machine learning models based on support vector machine (SVM) and random forest (RF),and compare their prediction performance with existing severity score SAPS system. Performance metrics selected for measuring model performance was important to assess the prediction ability of trained model and during hyperparameter optimization. The goal of this capstone project is to come up with hospital mortality prediction models that rely on a broad spectrum of medical features at admission and provides more favorable mortality prediction performance when compared to existing severity scoring system.

Metrics

Because the dataset is slightly imbalanced (~12% in hospital death), a simple accuracy score might not be sufficient to characterize classifier performance. That is, we will already have a 88% accuracy if we classify every patients as survivors after hospital admission despite the fact 12% died during hospital stay. Therefore, a combination of several performance metrics are reported on calibration models.As discussion in the ML nanodegree course, I opt for several metrics that better characterize a skewed dataset. Here I calculated precision, recall, f beta score, and receiver operating curve (ROC) with area under curve (AUC). Precision is defined as:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Recall is defined as:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

The F score is a weighted average of precision and recall. Here we selected $F_{0.5.}$ The reason is that we want to direct our limited medical care resouces to patients needing them most, thus we value precision over recall by not sending more patients to ICU units than what was actually needed. $F_{0.5}$ is defined as:

$$F_{0.5} = 1.25\frac{precision \cdot recall}{0.25 \cdot precision + recall}$$

Because we think that existing severity score system only explored a small fraction of features (~17 features) in the available dataset and its diagnostic performance is mediocre. We expect to come up with better predictive model with higher precision and mine the dataset by including more features (~40). The ROC and AUC are essential tools to summarize overall performance of different classifiers and relative performance when comparing to the benchmark model [7].

II. Analysis

Data Exploration

The MIMIC-III is a freely available database that was created and maintained by the Laboratory for Computational Physiology at Massachusetts Institute of Technology (MIT) and the department of Medicine at the Beth Israel Deaconess Medical Center (BIDMC). Because sensitive patient information was involved, anyone requesting access will have to go through training courses on respectful usage of clinical data before permission is granted. To set up a local copy of the MIMIC-III database, I downloaded the raw csv files from https://mimic.physionet.org/ and compiled the relational database using published sql scripts at https://github.com/MIT-LCP/mimic-code. The setup process took ~2 days to run on a home laptop computer. Once completed, the entire postgres database took up roughly 50 GB of hard drive space. Indexing the tables enabled faster query, but also took significant amount of time to set up initially. To calculate commonly used severity scores, another set of sql scripts were used to generate those scores such as SAPS as materialized view tables.

To generate the data required for the capstone project, sql script need to be written to extract medical records relevant to the classification problem. In general, data was pulled from four different sources, patient history including vital data, laboratory analysis results, e.g. glucose, platelet, charted event data, and output data, the urine output. Some of the fields in charted event data and output tables contains lots of data and need to be aggregated. Here I took average for those features as the aggregated values. A sample of 4 extracted data entries is shown in Table 1.

Table 1. Sample dataset for some selected features

subject_id	hadm_id	icustay_id	age	icu_los	heartrate	chloride	wbc	glucose
3	145834	211552	76	6.0646	92.37647059	112.5	12.53333333	154.7272727
4	185777	294638	47	1.6785	90.35483871		9.442857143	226.75
6	107064	228232	65	3.6729	86.54945055	100.8	13.58947368	124.7727273
9	150750	220597	41	5.3231	85.99481865	103	13.5	153.25

Table 2. Statistics for extract medical data

	count	mean	std	min	25%	50%	75%	max
male_flag	51907	0.56	0.5	0	0	1	1	1
emergency_admission	51907	0.86	0.35	0	1	1	1	1
ethnic_other	51907	0.13	0.34	0	0	0	0	1
ethnic_asian	51907	0.02	0.15	0	0	0	0	1
ethnic_hispanic	51907	0.03	0.18	0	0	0	0	1
ethnic_black	51907	0.1	0.29	0	0	0	0	1
ethnic_white	51907	0.72	0.45	0	0	1	1	1
hospital_expire_flag	51907	0.12	0.32	0	0	0	0	1

age	51907	74.35	54.71	18	52	65	77	310
icu_los	51907	4.21	6.2	0.04	1.23	2.17	4.24	173.07
hosp_los	51907	11.31	13.18	-0.77	4.25	7.36	13.35	294.66
heartrate	51907	84.79	13.88	29.92	75.14	84.12	93.82	163.88
sysbp	51899	120.72	16.31	48.26	108.98	118.92	131.06	215.83
diasbp	51899	61	10.44	22.56	53.81	60.07	67.21	127.1
meanbp	51904	78.68	10.72	28.5	71.2	77.53	85.1	140.56
resprate	51882	19.13	3.63	7.33	16.55	18.7	21.24	42.69
spo2	51884	96.91	2.22	46.43	96.04	97.2	98.21	100
urineoutput	49803	8297.2	24977.07	-34950	1990	3975	8305.5	4558660
urea	51796	25.73	18.5	1.42	13.6	19.67	31.63	232.04
magnesium	51267	2.02	0.25	0.6	1.87	2	2.15	8.98
calcium	49919	8.47	0.61	2.6	8.08	8.45	8.83	18.1
aniongap	51688	13.63	2.83	3	11.86	13.25	14.86	49
albumin	33065	3.15	0.66	1.05	2.7	3.13	3.6	5.7
bands	19525	3.37	5.9	0	0	1	4	69
bicarbonate	51697	25.4	3.88	6	23.33	25.41	27.38	52
totalco2	37329	25.8	5.17	1	23.06	25.54	28	63.5
carboxyhemoglobin	1610	1.92	1.63	0.1	1	1.4	2	25
bilirubin	35351	1.45	3.43	0	0.4	0.6	1.05	54.24
creatinine	51796	1.39	1.41	0	0.72	0.95	1.39	25.31
chloride	20908	104.63	6.13	3.8	101.29	105	108	198
hematocrit	51806	31.94	4.64	12.86	28.64	31.09	34.68	68.54
hemoglobin	51788	10.75	1.65	3.72	9.59	10.46	11.71	20.98
lactate	36399	2.18	1.56	0.3	1.3	1.8	2.5	24.8
o2flow	7930	9.29	12.43	0.4	3	4	10	70
platelet	51786	237.46	113.07	5.67	164.5	220.64	290.35	1551.39
potassium	51803	4.12	0.41	1.6	3.86	4.08	4.33	7.8
ptt	49732	37.23	14.98	16	27.2	31.8	42.58	150
inr	49871	1.43	0.62	0.6	1.11	1.25	1.49	32.4
pt	49866	15.32	4.72	8.57	13	13.95	15.76	150
sodium	51789	138.57	3.59	97	136.67	138.67	140.6	174.8
tidalvolume	16104	558.2	152.41	1	473.75	540	630	6300
bun	51796	25.73	18.5	1.42	13.6	19.67	31.63	232.04
wbc	51774	10.92	7.41	0.1	7.81	10.08	12.8	467.67
tempc	51843	36.89	0.65	29.2	36.5	36.85	37.24	41.1
glucose	51855	133.21	38.77	29.5	110.31	124.5	144.75	745.09
saps	51907	18.01	5.23	0	14	18	21	44
sapsii	51907	35.02	14.29	0	25	34	43	118

There are 51907 data entries and 51 fields with first three being identifiers for hospital admission, and ICU admissions. This dataset included patients admitted to Beth Israel Deaconess Medical Center (BIDMC) between 2001 and 2012. Of those included in this study, 56% were male. Over 85% hospital admissions were emergency ones. Majority of admitted patients were ethnic white. Roughly 12% of admitted patients eventually died during hospital stay. Average age is 74 years. Typical hospital admission last 11 days, while patients on average stay at ICU for 4 days. As mentioned, There are on average 4579 charted observation and 380 laboratory measurements for each hospital admission, which has to be reduced to average for each category to be utilized in this study.

The ethnic background information was a categorical variable with many different texts representing different races. I wrote sql script to reduce the categories into 5 distinct variables by implementing one-hot encoding. It was noted in the published paper on MIMICIII that patient age was intentionally offset to 300 for those aged 89 and older (total 2638 patients). I felt the older patient is valuable information, and thus decided to retain those patients in the dataset. There is only a fraction of those patients, thus, they were given a random age between 89-99[8]. A couple variables contains limited entries ('carboxyhemoglobin','o2flow','tidalvolume') or have null values throughout ('ventilationrate'). Therefore, it is decided to drop those columns. The remaining dataset mostly contain valid data values across all entries. However, some of them still possess significant number of missing values such as chloride (~20000 missing values). However, they seemed to be important features not to be directly discarded. Thus, I decide to retain those features in the dataset.

Exploratory Visualization

With this humongous amount of data, it is not easy to fully comprehend it by just looking at the summary statistics in Table 2. The visualization gave better sense regarding the distribution of the data and how features are related to each other and the target feature to be predicted. In figure 1, the scatter matrix of all 35 continuous numerical features is shown. The diagonal are probability density curve for each variable and at the each grid where two different features intersect, we have a scatter plot for the two features. As is seen in Fig.1, lots of features, i.e., glucose, white blood cell count (WBC), etc. have a relatively skewed distribution. So there might be lots of erroneous measurements that are recorded. However, without field expert knowledge, it is difficult to judge the validity of such claim. With some of the data produced either by ICU monitoring machines or certified bio-labs, however, the chance of wrong records at such magnitude is relatively low. Therefore, a more likelihood is that those data could be important data points that really correlates with patient sickness. Thus, outlier removal should be exercised with extreme cautions. It was also evident that some of the features are highly correlated, e.g., urea and bun, hemoglobin vs hematocrite. All other features did not show strong correlation with each other. Thus, inclusion of each of those features would hopefully bring in a little extra information to help us to piece the puzzle together more completely.

Despite the relevancy of all medical features to patient condition and potential for survival, not all features contribute equally to the successful prediction of mortality rates. Thus, some of the included features might be more important compared to others. Therefore, visualizing some of the features to find out if their distribution present a difference between patients who eventually discharged from hospital versus those who unfortunately died will give a rough idea as to what are those factors that play significant role in determining in-hospital mortality. Here, I picked age and platelet to showcase their importance to hospital mortality prediction. As is seen in Fig.2(a), patients eventually died during their hospital stay tend to be from older age population as compared to those survived. Center mean tend to bias towards people 80 and older in those deceased as compared to center mean of around 65 for those survived. For platelet, more survivors have a platelet value above 200 as compared to those deceased. Therefore, the two features may both be highly useful in predicting patient outcome.

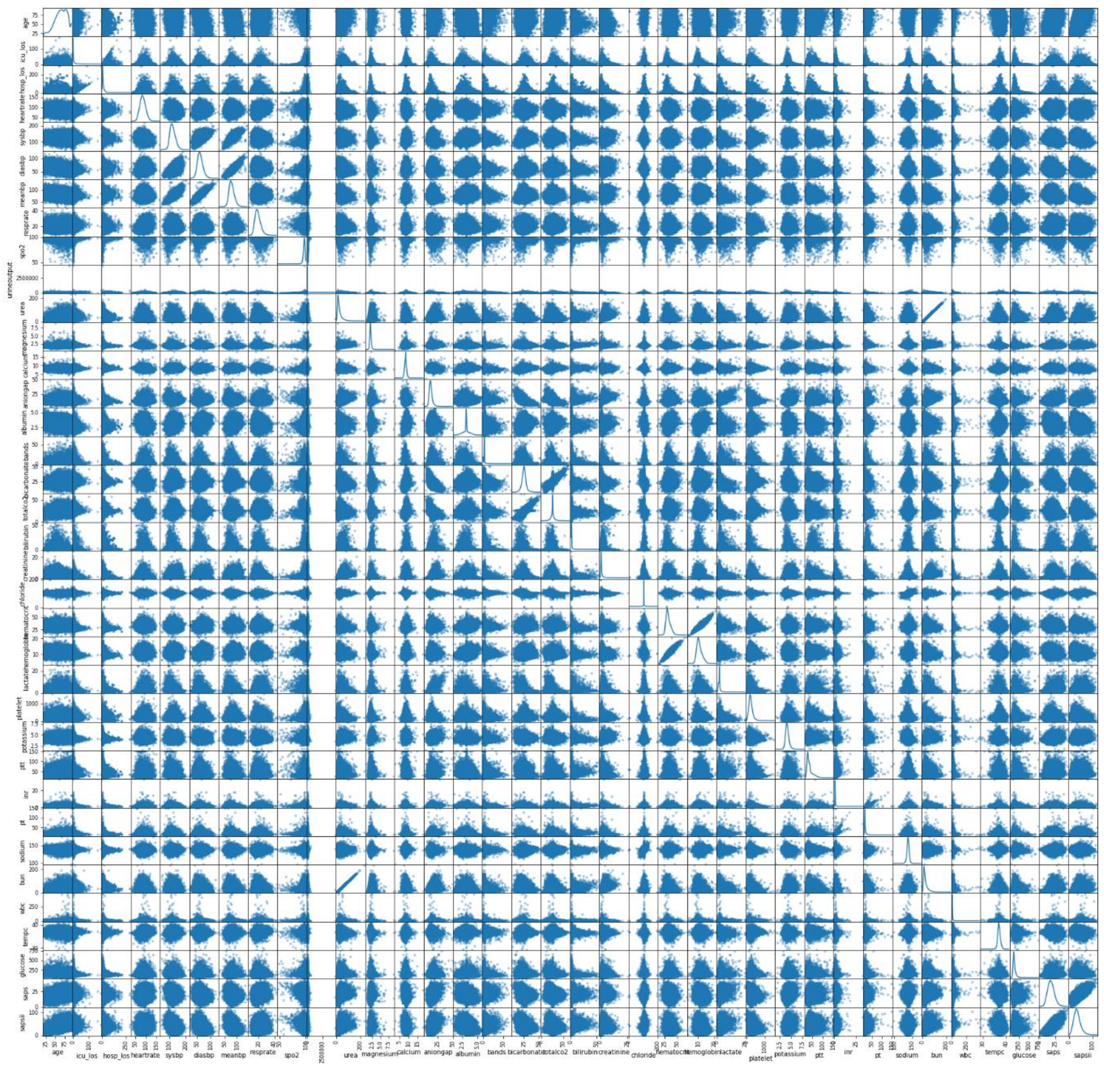


Figure 1.Scatter matrix of all numeric features used in mortality prediction

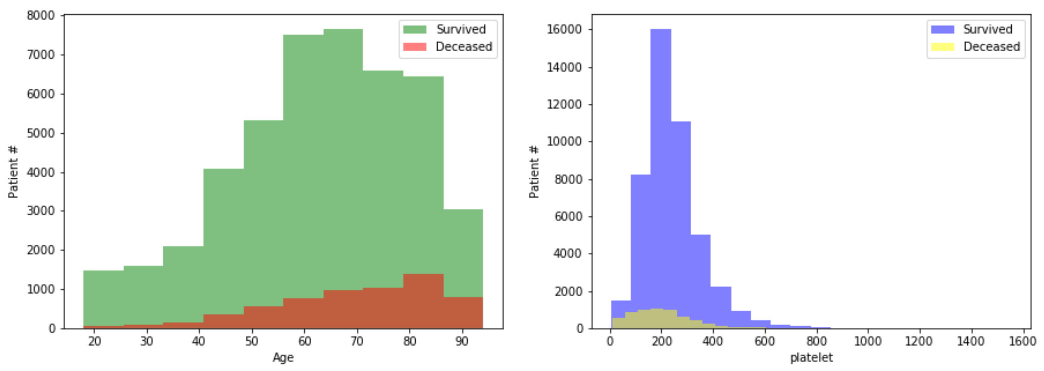


Figure 2. Histogram comparison between discharged and deceased patients

Algorithms and Techniques

For SVM classifier, the parameters to be tuned include C, the penalty term, and gamma, the kernel coefficient. Penalty term tells how tolerant SVM is towards making classification errors, i.e., larger C means smaller margin hyperplane. Low Gamma value generally means far reaching values influence the decision boundary and thus tends to bias the classifier. The kernel function could also be radial basis function, linear, or sigmoid. Since we are dealing with unbalanced dataset, I can also adjust the class weight by increasing relative weight to the positive class, i.e., the hospital deceased group. Because we are dealing with a large dataset with relatively high dimensionality (>40) with missing values, I need a machine learning method which can classify with a few samples (the support vectors). The SVC also offer nonlinear kernel functions to take away some of nonlinearity in data, which is expected as seen in data visualization section. In summary, a SVM classifier excels in all aforementioned areas is a good candidate to handle all aspects of the MIMIC-III dataset. The downside is that it is computationally expensive to train and selected features are not explainable.

Random forest classifier have a lot more parameters to tune, which include the number of estimators, max depth, min samples per leaf, etc [9]. Some of the parameters that matter most are: max_features, which dictates how many features to use on each tree; number of estimators, i.e., the number of trees to build; and minimum sample per leaf [10]. Despite the number of tunable parameters, it usually generalize well even with default parameters. When tuning, the RF classifier is much faster to train because its computation can be parallelized. The reason that RF is selected is because of its good generalization performance as an ensemble tree method. At the same time, RF classifier has lots of tuning parameters, making it highly adaptable. The RF method in scikit learn also can accommodate unbalanced class data, thus giving it another advantage. In addition, the RF model in scikit learn can list the set of features with most discriminant power, which give insights to how medical data can be directly related to mortality prediction.

I have also tested the gradient boosting tree (GBT) classifier. The original idea of gradient boosting was introduced by Friedman [11]. In general, boosting expects to come up with much stronger classifiers with an ensemble of weak learners, whose performance is expected to be at least slightly better than random guessing. Training was dense on those sample points where successful classification was found to be difficult. During training, those samples that are difficult to classify got increasing larger weights until a satisfactory model was produced. In gradient boosting, stage-wise additive models were expanded to the existing classification tree, thus overall prediction accuracy is expected to improve by inclusion of weak learners. In scikit learn, the gradient boosting classifier have several parameters to tune, I.e., n_estimators, learning rate, max_depth, min_samples_split.

Because all classifiers can handle high-dimensional data well to some degree, no data dimensionality reduction is performed. Thus, the extracted MIMIC-III features were fed directly to the classifiers for training. The dependent variable, i.e. hospital mortality, was binary coded (0 for survival and 1 for death) and used as the target variable for model training. Once parameters for an optimized model is ready, the best performing models are used to predict the independent test set and subsequently generate all performance statistics such as F1 score, AUC values, etc.

Benchmark

The benchmark model utilize a well-established severity score Simplified Acute Physiology Score (SAPS) to calculate the predicated mortality probability for the patient. The calculation of SAPS score is based on a collection of 14 physiological variables and their deviation from normal [11]. Most of the parameters included in SAPS calculation, such as age, heart rate, temperature were also included in the dataset that I used to train the machine learning algorithms. Selection of those variables for SAPS calculation was partially empirical with inputs from medical practitioners and validated in hospitals. Many published literatures have extensively utilize the SAPS or similar severity scores as the benchmark to assess the performance of their machine learning models [12]. To calculate the predicted mortality probability, the following equation is used [13]:

$$Predicted\ Death\ Rate = \frac{e^{(Logit)}}{1+e^{(Logit)}}$$

Where Logit=7.7631+0.0737*SAPS+0.9971*ln(SAPS+1)

III. Methodology

Data Preprocessing

The database authors have intentionally offset patients' age to 300 when they are aged 89 years and older. There are 2368 entries (<0.5% of total sample size) with age at 300. I felt the age group is important to this classification task, so I have assigned these patients a randomized age between 89 and 99 on the assumption that age beyond 89 is significant factor in mortality prediction. Still, the data distribution as is seen in Fig. 1 seemed to be quite skewed and some with strong kurtosis. Therefore, it makes sense to remove some data at the further ends of the tails. By applying the interquartile range (IQR) rule directly across all features almost resulted in exclusion of ~30,000 data entries (mostly caused by some features such as chloride). Therefore, natural log transform was applied before IQR rule, but still significant number of entries were discarded by IQR rule. A published study on MIMIC-III stated that outlier exclusion should be exercised with extreme caution without prior expert knowledge in the medical field. The recommendation is to not exclude or minimize exclusion of any measurements generated by machines or electronic health record system because there will be rare error. What is more important is that those extreme values contained information relevant to patient morbidity and should not be discarded because they did not fall within normal distribution. Thus, outlier removal was only conducted on a few aggregated features. The ethnical background feature is categorical, thus I wrote sql script to implement one-hot encoding to make five distinct features. For other numerical features, standard scaler feature transform is applied to assign equal weight. Because both SVM and RF classifiers can handle nonlinearity in data, no normalization feature transform was performed. For missing values in any of those fields (generally much less than 10% of total entries for most features), the missing values were replaced with the mean value for that field (column).

Implementation

Before training, all administrative data, e.g., identification number, severity scores, and target variable (hospital mortality) were removed from the predictor matrix. Then the dataset, both predictor and dependent variable, were split into training and test using stratified method according to the dependent variable. Thus, we can ensure that both training and test set have the same proportion of positive and negative data entries. For benchmark model implementation, it is a straightforward mathematical formula, which is directly written inline with other codes. A section of code to plot the ROC curve and calculate the AUC followed benchmark model calculation. After dropping all administrative information and SAPS scores. The predictor matrix was scaled using the standard scaler function from scikit-learn's preprocessing module. The splitting ratio is 3:1, thus 75% was used for training and 25% was held out for independent test. A total of 36120 data entries were kept for training and 12041 samples for test. After transformation and scaling, there were 41 features that were included for setting up classification models.

Before actually moving on to fine tuning of classifiers, I was interested to find out how well those classifier perform with default parameter settings, which could be used as a baseline metric to see how much improvement we could get after fine tuning our classification models. Thus, I set up both a SVM classifier and RF classifier with default parameter settings. Then I fit the model to scaled training data set. Then, test their performance using independent test data. This was accomplished with the following lines of codes:

```
SVC_native=svm.SVC(probability=true)

RF_native=RandomForestClassifier()

svcn_model=SVC_native.fit(X_train_scaled,y_train)

rfn_model=RF_native.fit(X_train_scaled,y_train)
```

To be able to compare across different classifiers using performance metrics as stated earlier, I have also written a function to summarize each classifier by taking in them as a dictionary. Then, all performance metrics were put into a dataframe and printed on the notebook. Since the reporting performance metrics remain the same for classifiers with default parameters and classifiers to be optimized, the AUC, accuracy, precision, recall, and $f_{0.5}$ score were computed as columns and different classification methods were listed in two rows. The f score is probably the one metric that need to specify the alpha value. Other metrics are not modified. The script for the summary statistics function is as follows:

```
def stat_sum(x_test,y_test,classifier_dict):

    sum_table=pd.DataFrame(data=None,
index=classifier_dict.keys(),columns=['Accuracy','Precision','Recall','f-score','AUC'])

#     print(sum_table)

    for key,classifier in classifier_dict.items():

        y_score=classifier.predict_proba(x_test)

        y_predict=classifier.predict(x_test)

        fpr, tpr,_=roc_curve(y_test,y_score[:,1])

        roc_auc=auc(fpr,tpr)

        accu=accuracy_score(y_test,y_predict)

        prec=precision_score(y_test,y_predict)

        recall=recall_score(y_test,y_predict)

        fs=fbeta_score(y_test,y_predict,0.5)

        stats=[round(elem,2) for elem in [accu,prec,recall,fs,roc_auc]]

        sum_table.loc[key]=stats

    print(sum_table)
```

So, each time I re-train the models, I will calculate new performance statistics for the new models and print them. With default parameters for SVC and RF classifiers, I have the following performance statistics:

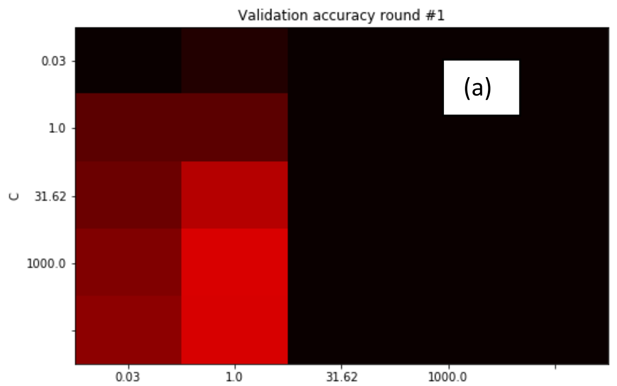
Table 3. Performance metrics for SVC and RF with default parameters.

	Accuracy	Precision	Recall	f-score	AUC
SVM	0.93	0.83	0.43	0.7	0.92
Random	0.92	0.84	0.38	0.67	0.9

The metrics values are already encouraging compared the relatively simple SAPS logistic model. The SVM seemed to have the upper hand with default parameters, but we should remember that RF has more tunable parameters and might prove to be more accurate upon further parameter adjustment. Based on the AUC values, both SVM and RF already have won over the SAPS method. However, it should be noted that the recall values are somehow fairly poor, i.e., fair amount of patients dismissed as possible survivors actually died in hospital. Something needs to be improved.

Refinement

For fine tuning of SVM classifier, I complied parameter grids that is to be quickly searched on first run and then migrated to a more targeted range for fine tuning. As discussed, the important parameters are C, the penalty term, and gamma, the distance-influence term. Smaller C means a classifier more tolerant of misclassification (high bias) and smaller gamma means high influence from support vectors near the boundary (high variance). Accordingly, the initial search range is set as from 10^{-3} to 10^{3}, a common range to search for good combinations of C and gamma, interpolated at 5 different levels each. Radial basis function was tested as the kernel function. To deal with the slightly unbalanced data, class_weight was also assigned to be higher than 1 for the positive class. Two values (3,9) were programmed to be tried. After assembling all parameters to be tuned, I put them into a dictionary variable,'parameters_svm'. For hyperparameter tuning, I chose grid search in a hope to find the optimium combination among all tunable paramters to get the best performing SVM model for prediction. I selected 5 fold cross validation and set 16 concurrent jobs to run in parallel. The performance metric is $f_{0.5}$ score. The grid search process is significantly slow, but fortunately I only have 125 combinations to try. If I elect to search at 10 instead of 5 intervals for C and gamma, I can easily get stuck with over many hours of computing time even on a high-performance computer for SVM optimization. After initial quick scan through the C vs gamma range, a good idea about where the optimum could be located could thus been derived (Fig.3 (a)). In this project, the optimum combination of C and gamma clearly should have high C values and yet small gamma, thus more model variance would improve model performance. Thus, during second run, the scan range was shifted to the new combinations of C and gamma (C: 10^{-1}-10^{4}; gamma: 10^{-5}-1) in hope for obtaining better generalized prediction performance (Figure 3(b)). Somehow the new scan range is a refined search of the previously identified better range, but with limited improvement, partially due to the prohibitive computing cost associated SVM training during grid search process.



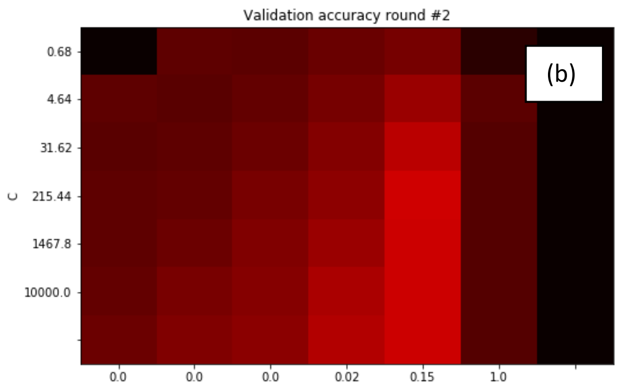


Figure 3. SVM grid search results. (a) C and gamma scan range (10^{-3} to 10^{3}); (b) C and gamma scan range (C: 10^{-1}-10^{4}; gamma: 10^{-5}-1)

The summary statistics is shown in Table 4. As is shown in the table, exhaustive grid search did not improve the classifier performance, except the recall values, but rather resulted in suboptimal solutions. This is mainly due to a rather coarse grid search while the default C (1) and gamma (0.02)was already somewhere in the optimal parameter scan range.

Table 4. Performance metrics for SVC with different parameters

	Accuracy	Precision	Recall	f-score	AUC
SVM	0.93	0.83	0.43	0.7	0.92
SVM grid search range 1	0.92	0.65	0.6	0.64	0.91
SVM grid search range 2	0.92	0.63	0.62	0.62	0.91

For tuning RF classifier hyperparameters, similiar approach was used for selecting parameter combinations except that I chose to use randomized search cv instead of a thorough grid search in anticipation of the huge number of models to try (36,000*5=180,000 !). Here, I selected 5 RF parameters to optimize, i.e., number of estimators, max_features selection function, maximum depth, minimum number of samples required for a split, and minmum number samples per leaf. I randomly selected 1000 out of the total 36,000 possible combinations to try. The class weight was also listed as a parameter to be changed during training. The parameters variables was put into a dictionary and fed into the RandomSearchcv. Other than that, similar setup like SVM optimization was applied to RF classifier parameter tuning. Due to the parallel computing feature for RF classifier training, it trained much faster, taking 529 mins to complete, while 125 SVM model fits already took 441 mins to finish. What is more appealing for RF classifier was that the model performance was generally improved without sacrificing the precision while there was an enhancement to the recall value.

Table 5. Performance metrics for SVC and RF after hyperparameter tuning

	Accuracy	Precision	Recall	f-score	AUC
SVM	0.92	0.65	0.6	0.64	0.91
RF	0.93	0.8	0.5	0.71	0.94

The excellent performance by RF classifier consolidated my belief in tree-based classifier, and I think the gradient boost tree classifier might give the best performance. Similiar parameter tuning procedures was applied to GBT classifier, but I chose to use grid search because there were less number of classifiers to try. For GBT hyperparameter tuning, I chose learning_rate (3 values), n_estimators (50), max_depth (3), and min_samples_split (3), to tune. All are given a relatively reasonable ranges, I.e., 100 to 1000 at interval of 50 for n_estimators. Somehow, GBT classifier was also not easy to train and took considerable amount of time to complete the entire session (880 mins). On the other hand, the independent test set results were the best among all classifiers that have been trained so far (Table 6). As is seen in the table, both optimized RF and GBT classifiers offered satisfactory performance. This is especially true for GBT classifier, which provided improvement in both precision and recall scores at the same time, where recall might be more important in mortality prediction.

Table 6. Performance metrics for SVC and RF after hyperparameter tuning

	Accuracy	Precision	Recall	f-score	AUC
RF default	0.92	0.84	0.38	0.67	0.9
RF optimized	0.93	0.8	0.5	0.71	0.94
GBT optimized	0.95	0.86	0.62	0.79	0.96

IV. Results

Model Evaluation and Validation

Finally, I came up with two classes (SVM and tree based) and three different types classification (SVM, RF, GBT) models that have the best generalization performance. In general, the performance improvement over the course of training and optimization aligned with my expectations especially regarding GBT classifier. Tree based classifiers are awesome in binary classification problems with generalized prediction performance that often superseded other types of classifiers such as SVM. On the other hand, SVM classifier also performed reasonably well except that they are not so easy to train and interpret. Both final models were obtained through thorough search through the hyperparameter space to find the optimal combination of parameters that have the lowest cross-validation errors and highest $F_{0.5}$ scores. The SVM generally works well with a relatively high C and lower gamma, which means model with lower bias and high variance. This was reasonable because the unbalanced data required a high variance classifier to offer good prediction. The models were then validated by the initial independent test set.All performance metrics as defined in the method section were calculated. The performance of final classifier was judged mainly by both F value and AUC for ROC. I implemented stratifiedshufflesplit to enable that the data, during training, were shuffled so that during different cross validation loops, data set aside for training and validation are as different from other folds as possible. Stratified split also ensure that samples for each class could preserve the percentage of their classes respectively. Thus, this strategy helped introduce perturbation during training, making it more robust to small changes in the training dataset. The resulting classification model was tested on independent test set, which has been set aside at the beginning of training session and never been seen by the model during the training. The test set accounted for 25 percent of the total data and was 11.2%, the same ratio in the whole dataset. With multiple classifiers tested on the independent dataset, the prediction performance roughly agreed with each other. Thus, I think the performance statistics generated by predicting patient mortality using the final models are reliable and adequately summarized the generalization performance of calibrated models.

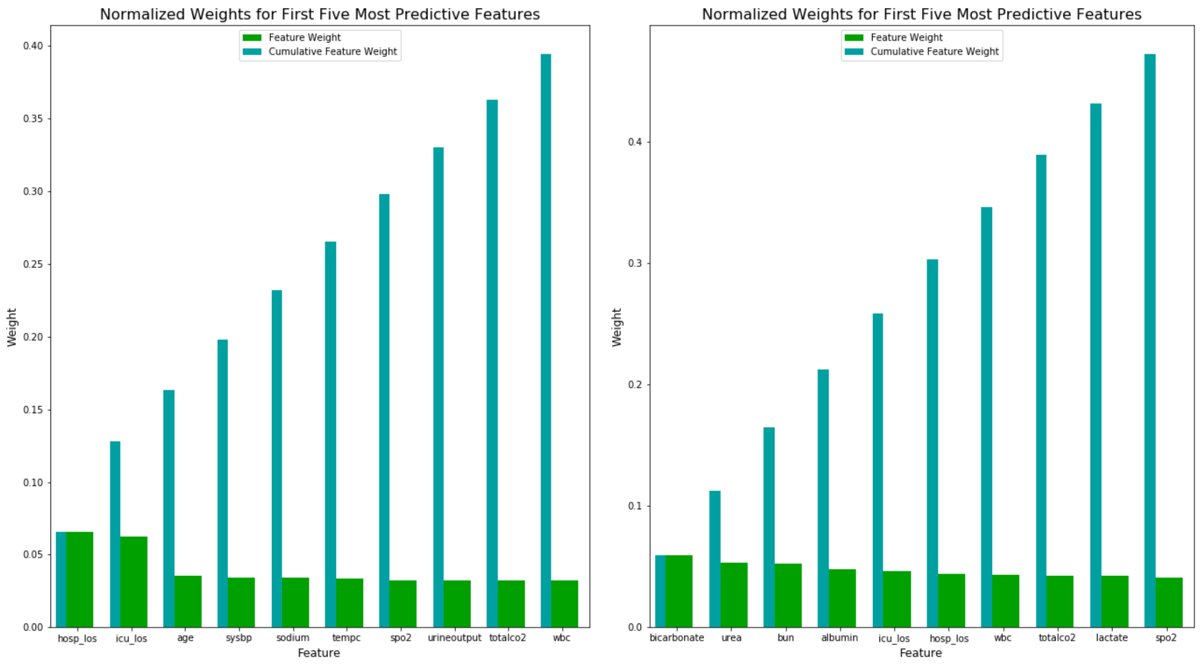


Figure 4. important features for classification (a) top 5 for GBT; (b) top 5 for RF

Another good thing about tree based classifiers are that they provide feature importance measures, which helped to identify those relatively important features, so that we could potentially come up with a more succinct model without sacrificing too much accuracy. As is seen in Figure 4(a),GBT picked out the parameters that contributed most to mortality prediction. It made lots of senses to me since hospital death did intuitively highly correlate with length of stay in hospital, length of stay in ICU unit, and age. The RF classifier also identified the ICU length of stay as one of the top five features among other four biometric parameters. Although there was not many overlapping features in the top 5, (which explained the difference between GBT and RF), there was more overlap between important features among the top tens. As was observed in the feature space, there were generally two categories of features that were critical in predicting patient mortality, i.e., vita signs such as length of stays, age, body temperature, and blood analysis parameters such as, white blood cell,SpO2,bicarbonate levels. This was quite illuminating results because the doctors could rely on doing a blood analysis targeting those important features and combining that with patient vital record to make adequate conclusions.

Justification

For SAPS scoring model, the AUC is 0.75, which is fair with such a simple model.This number agreed well with what was reported in literature using the MIMIC dataset and SAPS scoring system [15]. When compared to the machine learning models developed in this capstone project, one can clearly see that there is striking difference in prediction performance. Without parameter tuning, the SVC and RF models already prove to be advantageous in terms of AUC, a summary statistics regarding the diagnostic capacity for different classifiers. I believe the results obtained in the project is a much better solution compared to the conventional SAPS based scoring system. First, it is much more powerful compared to the SAPS based model in terms of AUC values. Second, the GBT offers a fairly simple and yet highly accurate model just base on a much smaller subset of features. We have amazingly great performance statistics for independent test (see table 6)

V. Conclusion

Free-Form Visualization

The ROC curves for all trained classifiers were plotted in Figure 4. As is shown in the figure, even classifiers with default parameters (SVC and RF) have considerably better performance compared to the conventional SAPS based severity scoring system. The best performing classifier were tree based classifers after hyperparameter optimization. Hyperparameter tuning in SVM unfortunately did not resulted in significant improvement in generalized prediction performance, partially owing to the fact that the default parameters were already within the approximate range of optimal combinations of C and gamma. Because of the costly computation associated with SVM classifier tuning, a rather crude grid search was performed, which also might have contributed to suboptimal solutions for the SVC classifiers. Tree based classifiers proved to be a great success. Especially, GBT classifier offered the highest AUC values along with best performance metrics.

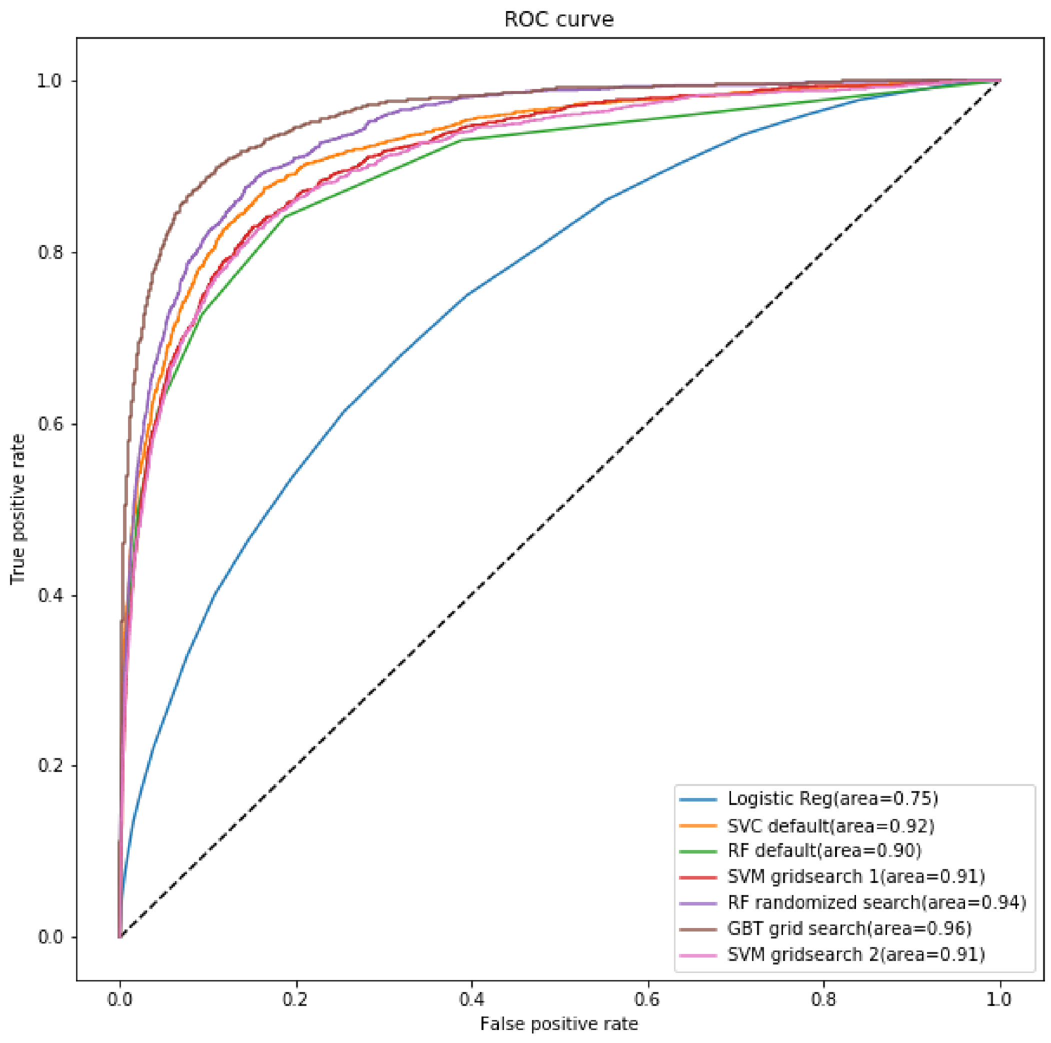


Figure 4. ROC curves for all classifiers.

As is discussed in results section, the GBT classifier outperformed other classifiers in independent test and offered a more simplified solution if a selected subset of features was chosen for patient mortality prediction. Here, I shrinked down the number of features included for GBT prediction models for the top 5 and 10 important features. As is seen in Table 7, reduced model with top 5 selected features resulted in considerably degraded and yet still good model performance. This was like simulated situations where doctors could make fairly accurate judgement on patient mortality by recording the few vital biometrics, e.g., blood pressure, hospital length of stay, age, etc, and feed that into the GBT model. If the doctor want more accurate prediction, a simple blood test with a few blood properties, e.g., WBC and bicarbonate, would greatly improve model performance resulting in better prediction. This has rather practical implications in terms of maximizing utilization of hospital resources. With a precision as good as 0.86, the doctors could transfer those patients to ordinary care units with only 14% of those would mostly likely be survivors, although remaining in ICU might improve their chances. A Recall rate of 62% suggested that still less than 40% of those destined to die in hospital would remain to receive ICU care, a much less aggressive approach for those with unfavorable fates. With the current simplified GBT patient mortality prediction, there would be much more cost-saving based on simply patient vital signs and a blood test. A great alternative to better healthcare for all without further investment.

Table 7. Performance metrics for GBT with reduced feature space

	Accuracy	Precision	Recall	f-score	AUC
GBT	0.95	0.86	0.62	0.79	0.96
GBT5	0.91	0.65	0.38	0.57	0.86
GBT10	0.93	0.77	0.5	0.7	0.93

Reflection

This capstone project gave me great exposure to lots of things that are outside of the realm of machine learning. The machine learning actually is an integral and yet not the most difficult part of the overall process. I chose to work on hospital mortality prediction because my interest in big medical data and how electronic health record (EHR) could be best utilized to derive valid conclusions helpful to maximize healthcare resource utilization and better patient outcomes. Coming from biomedical engineering background, I always have an interest in ways to improve our healthcare system, especially with a rapidly aging population. Our beloved ones will eventually require medical attentions at some point and how medical practitioners could be best informed decisions is critical. Thus, taking online courses in health bioinformatics and reading through dozens of papers enabled me to get a good understanding about how machine learning, big data is going to help with improving our healthcare system. It was quite illuminating when I finally be able to connect some of the dots and being able to work with the MIMIC-III database.

Setting up and query the MIMIC-III database was a great challenge at first because I don't have much database programming experience. Although I dealt with lots of data, mainly from all sorts of instruments, getting started to think of metadata in a relational database (RDB) framework was relatively new to me. Thus, learning the RDB concepts and practicing to manage and set up database has been quite enjoyable. After going through many tutorials about working with postgresql database and reading through some of the sql scripts shared by MIT lab on github, I became more confident about working with RDB. Setting up of MIMIC-III has been a struggle at first since there is not very thorough documents about its installation. But after working with it for a while, I began to grasp the concepts more easily and can even do a bit of changes myself. Query the data to come up with the dataset for mortality prediction was another challenge that I got myself into and I happily survived its test. There are so many different types of variables, e.g., integer, floats, list of floats, strings, and some of them must be aggregated into mean or over several different code fields. Eventually, I managed to get an handle on this thing and was able to write sql script to extract all the relevant data.

Certainly, getting the data out from the database was just the beginning part of the machine learning project. I then immediately faced the problem of how to properly clean the medical data to prepare it for machine learning training. Entries of string has to be converted to binary codes based on one-hot coding rule. There are many more missing values that I originally expected. Thus, whether to leave out those entries or replacing it with values that did not significantly disrupt the data flow was important. It was also noticed that data distributions are highly skewness with strong kurtosis in some of the features. Therefore, whether and how to exclude those seemingly outlying datapoints was also a question. All the aforementioned concerns warrant thorough research and have to be dealt with case by case. After data cleansing, I have chosen to work with support vector machine and random forest classifiers for machine learning part. The reason that I think SVM would work was because it is tolerant of outliers and generalize well. As a tree based method, the RF approach is a straightforward method that usually works fine without much tuning, although lots of parameters are available for tuning. The classifiers already worked to a degree with default parameters. So I gathered the tuning hyperparameters and tried to perform grid parameter cross-validation search for optimal parameter set. This optimization process took painfully long to complete and I have to exit the python kernel multiple times after days of training because there is something new that I would like to try. So, it has not been completed after a week's non-stop computation. Therefore, it was important to decide well beforehand the parameters and search ranges, so that one don't get caught hand-tied in time-critical tasks.

Improvement

Due to the extremely lengthy training time involved in grid search cross validation, I have to restrict the search grid range to the most commonly used range, which already proved to take incredible amount of time to find a local minimum. Therefore, if possible, I would keep on improving the classifiers by drawing those heat maps for any two parameters, e.g., C and gamma for SVM, and figure out where the minimum is located. If it appeared that the best metric score is somewhere along the range boundary, I would probably start a new search in a different range closer to the best metrics point. Again, this is really computationally expensive process and may require better hardware support to make the algorithm converge faster.

Certainly, the possibilities with choices of machine learning methods are endless. Time and resource permitting, I would like to try some more interesting methods such as extreme gradient boost (xgboost) and deep learning neural network(DLNN). The tree-based xgboost outperformed many different machine learning algorithms in several Kaggle competitions. Of course, it has a whole lot of parameters to tune, which is as well computationally expensive to calibrate. One attractive training feature about xgboost is that it allowed multi-thread parallel computing at its kernel, which might be faster than the sklearn parallel computing. I am also quite fascinated by the use of deep learning methods to solve big data problem. The medical data is a structured big data problem, where I may see use of DLNN to be an overkill. In addition, I currently don't have access to the infrastructure to do the computing needed to train convolutional neural network more than a few layers big. So I will probably hold onto to that thought and revisit it later.

From the data point of view, I have selected a set of medical records that are believed to be most relevant to mortality prediction. Somehow, I may have left out information The SVM and RF classifiers are popular algorithms that would likely work in most cases and generalize well on new data, but it does not necessarily mean that they are the best choice for the MIMIC-III dataset.

Reference

[1] Centers for Medicare & Medicaid Services. National Healthcare Expenditure Fact Sheet.https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/NationalHealthExpendData/NHE-Fact-Sheet.html. Accessed on April.2nd, 2018.

[2] Kaiser Health News. IOM Report: Estimated $750B Wasted Annually In Health Care System. https://khn.org/morning-breakout/iom-report/. Accessed on April. 2nd, 2018.

[3] Johnson, A. E. W., Pollard, T. J., Shen, L., Lehman, L. W. H., Feng, M., Ghassemi, M., Mark, R. G. (2016). MIMIC-III, a freely accessible critical care database. Scientific Data, 3. https://doi.org/10.1038/sdata.2016.35

[4] Gall JR, Lemeshow S, Saulnier F. A New Simplified Acute Physiology Score (SAPS II) Based on a European/North American Multicenter Study. JAMA J Am Med Assoc. 1993;270(24):2957-2963. doi:10.1001/jama.1993.03510240069035.

[5] Rogers J, Fuller HD. Use of daily Acute Physiology and Chronic Health Evaluation (APACHE) II scores to predict individual patient survival rate. Crit Care Med. 1994;22(9):1402-1405. doi:10.1097/00003246-199409000-00008.

[6] Keegan MT, Gajic O, Afessa B. Severity of illness scoring systems in the intensive care unit. Crit Care Med. 2011;39(1):163-169. doi:10.1097/CCM.0b013e3181f96f81.

[7] Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 2012;12:2825-2830. doi:10.1007/s13398-014-0173-7.2.

[8] Johnson AE, Pollard TJ, Shen L, Lehman LW, Feng M, Ghassemi M, Moody B, Szolovits P, Celi LA, Mark RG. MIMIC-III, a freely accessible critical care database. Scientific Data. 2016;3.

[9] Scikit-learn documentation on random forest classifier http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[10] Srivastava, T. 2015. Tuning the parameters of your Random Forest model. https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/

[11] JH Friedman. 2001. Greedy function approximation: a gradient boosting machine. Annals of statistics, 2001 - JSTOR

[12] Gall JR, Lemeshow S, Leleu G, et al. Customized probability models for early severe sepsis in adult intensive care patients. Intensive Care Unit Scoring Group. J Am Med Assoc 1995; 273: 644–50

[13] Rogers J, Fuller HD. Use of daily Acute Physiology and Chronic Health Evaluation (APACHE) II scores to predict individual patient survival rate. Crit Care Med. 1994;22(9):1402-1405. doi:10.1097/00003246-199409000-00008.

[14] Gall JR, Lemeshow S, Saulnier F. A New Simplified Acute Physiology Score (SAPS II) Based on a European/North American Multicenter Study. JAMA J Am Med Assoc. 1993;270(24):2957-2963.

[15] Celi LAG, Tang RJ, Villarroel MC, Davidzon GA, Lester WT, Chueh HC. A Clinical Database-Driven Approach to Decision Support: Predicting Mortality Among Patients with Acute Kidney Injury. Journal of healthcare engineering. 2011;2(1):97-110. doi:10.1260/2040-2295.2.1.97.