UNIVERSITY OF CALIFORNIA

Los Angeles

Imbalanced Binary Classification On Hospital Readmission Data With Missing Values

A thesis submitted in partial satisfaction of the

requirements for the degree

Master of Applied Statistics

by

Hui Zhang

2018

ABSTRACT OF THE THESIS

Imbalanced Binary Classification On Hospital Readmission Data With Missing Values

by

Hui Zhang

Master of Applied Statistics

University of California, Los Angeles, 2018

Professor Yingnian Wu, Chair

Hospital readmission is a costly, undesirable, and often preventable patient outcome of inpatient care. Early readmission prediction can effectively prevent life-threatening events and reduce healthcare costs. However, imbalanced class distribution and high missing value rates are usually associated with readmission data and need to be handled carefully before building classification models. In this paper, we investigate the prediction of hospital readmission on a dataset with high percentage of missing values and class imbalance problem. Different methods are applied to impute missing values in the categorical variables and numerical variables. In addition, SMOTE (Synthetic Minority Over-sampling Technique) and cost-sensitive learning are combined with different classification methods (LASSO logistic regression, random forest, and gradient boosting) to explore which one will yield the best classification performance on the readmission data. Total misclassification cost and area under ROC curve are used as evaluation metrics for model comparison. Our results show that the SMOTE method causes overfitting on our readmission data and cost-sensitive learning outperforms SMOTE in terms of total misclassification cost.

The thesis of Hui Zhang is approved.

Nicolas Christou

Jingyi Li

Yingnian Wu, Committee Chair

University of California, Los Angeles

2018

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. Introduction

Classification is one of the most common and important tasks in machine learning. A classifier can be learned from a set of training samples with predefined class labels, and can be used to predict new observations into said classes. The class label is usually discrete and finite, and the problem is known as binary classification when there are only two classes. Many classification modeling algorithms, such as logistic regression, decision tree, random forest, gradient boosting, support vector machine, and neural network, have been well developed and successfully applied to many application domains.

In this paper, we propose to develop, validate and assess classification algorithms that predict hospital readmission for individual patients. A hospital readmission is defined as admission to a hospital within a specified time frame after being discharged. Different time frames such as 30-day, 90-day, and 1-year readmissions have been used for research purposes. Readmission is a costly, undesirable, and often preventable patient outcome of inpatient care. High readmission rate reflects relatively low quality and also has negative social impacts on the patients and on the hospital. Nearly 20 percent of hospital patients are readmitted within 30 days of discharge, a $35 billion problem for both patients and the healthcare system. Avoidable readmissions account for around $17 billion a year [1]. In the hope of reducing hospital readmission rates, federal government is penalizing hospital based on 30-day readmissions being over certain standard rates. Therefore, 30-day readmission has become an important indicator for evaluating the healthcare effectiveness of a hospital.

The dataset we use in this paper is provided by Providence Health. It contains 552,816 observations and 116 predictor variables. The response variable *READMITIN30DAYS* is a binary variable with values 0 or 1 for each instance, 1 represents the discharged patient was readmitted within 30 days after the original admission, and 0 means the opposite. Although we have more than enough examples and predictors to train binary classification models, there are two main issues in this dataset: high percentage of missing values and imbalanced class distribution. In particular, the missing rates for some variables are even higher than 60%, and only 13.5% of the observations will be included in the analysis if we only use complete cases. For binary classification problem, imbalanced class distribution is characterized as having many more instances of one class (majority class) than the other (minority class). In the readmission dataset, approximate 87% of the examples are labeled as negative class, and 13% of the examples are labeled as positive class.

The impact of missing data can be serious, leading to loss of information, biased estimates of parameters, increased standard errors, and weakened generalizability of findings. Learning from datasets with imbalanced class distribution is also a difficult task since most learning algorithms are not designed to cope with a large difference between the number of cases belonging to different classes [2]. Standard classification algorithms usually perform poorly on imbalanced datasets because they focus on maximizing the overall classification accuracy, but in the imbalanced class scenario, high accuracy can simply be achieved by just predicting all instances as the majority class. Consequently, test samples belonging to the minority class are misclassified more often than those belonging to the majority class.

The rest of the paper is organized as follows. Following the introduction, Chapter 2 introduces some machine learning algorithms that are most commonly used for binary classification, including LASSO logistic regression, random forest, and gradient boosting. Chapter 3 provides a thorough discussion on the missing value problem and imputation algorithms. Chapter 4 presents a comprehensive study on the class imbalance problem, including the nature of the problem, resampling and cost-sensitive learning approaches as solutions to the class imbalance problem, and proper evaluation measures for classification with class imbalance problem. Chapter 5 analyzes and compares the performance of resampling and cost-sensitive learning methods on our readmission dataset, and reports our experimental results.

CHAPTER 2. Common Machine Learning Methods For Binary Classification

2.1 Logistic regression

The logistic regression model is one of the most common methods used to estimate the probability of a binary response based on input data. The key idea for logistic regression is as follows: Consider a dataset with n training examples, where $X_i^T = (x_{i1}, \cdots, x_{ip})$ consists of p predictors, $y_i \in \{0,1\}$ is the outcome or class label. The model parameter is $\beta = (\beta_0, \beta_1, \beta_2, \cdots, \beta_p)^T$. In the logistic regression, we assume $y_i \sim Bernoulli(p_i)$, i.e., $P(y_i = 1 \mid X_i, \beta) = p_i$, and we assume

$$logit(p_i) = log \frac{p_i}{1 - p_i} = X_i^T \beta \tag{2.1}$$

Then we have

$$p_i = sigmoid(X_i^T \beta) = \frac{exp(X_i^T \beta)}{1 + exp(X_i^T \beta)} = \frac{1}{1 + exp(-X_i^T \beta)} \tag{2.2}$$

where the sigmoid function is the inverse of the logit function. By default, a threshold of 0.5 is used to decide the predicted label of a new observation. Therefore, the final prediction is $\hat{y}_i = 1$ if and only if $P(y_i = 1 \mid X_i, \beta) > 0.5$.

For logistic regression, in order to learn β from the training examples, we usually maximize the likelihood function, which is

$$\prod_{i=1}^{n} P(y_i \mid X_i, \beta) \tag{2.3}$$

That is, we want to find β to maximize the probability of the observed y_i given X_i. The maximum likelihood estimate gives the most plausible explanation to the observed data. For $y_i \in \{0,1\}$, one can find that the log-likelihood is $\sum_{i=1}^{n} logP(y_i|X_i, \beta) = \sum_{i=1}^{n} [y_i X_i^T \beta - log(1 + exp(X_i^T \beta))]$.

And the loss function is defined as the negative log-likelihood

$$\sum_{i=1}^{n} L(y_i, X_i^T \beta) = - \sum_{i=1}^{n} [y_i X_i^T \beta - log(1 + exp(X_i^T \beta))] \tag{2.4}$$

Then we just need to find the β that minimize the loss function, which is the same as maximizing the log-likelihood.

2.2 Regularized logistic regression

Regularized logistic regression is usually used when we have many parameters to estimate, i.e., β is a high dimensional vector. In this case, we need to regularize β by adding a penalty term to the logistic loss function as the following equation

$$\mathscr{L}(\beta) = \sum_{i=1}^{n} L(y_i, X_i^T \beta) + \lambda \rho(\beta) \tag{2.5}$$

where $\rho(\beta)$ is a regularization function and λ is a regularization constant to be carefully tuned using a validation set or cross-validation. Popular choices of $\rho(\beta)$ include (1) ℓ_1 regularization, where $\rho(\beta) = ||\beta||_{\ell 1}$, which is the sum of the absolute values of the components of β. (2) ℓ_2 regularization, where $\rho(\beta) = ||\beta||_{\ell 2}^2$, which is the sum of squares of the components of β. A regression model that uses ℓ_1 regularization technique is called LASSO (least absolute shrinkage and selection operator) Regression and model which uses ℓ_2 regularization is called Ridge Regression. In the case of ridge regression, the effect of the penalty term is to shrink the coefficients that contribute most to the error. In contrast, in the case of LASSO regression, the effect of the penalty term is to set these coefficients exactly to zero.

2.3 Random forest

Random forest is an ensemble method for classification and has been shown to reduce the variance and prevent overfitting. Given a dataset $\{(X_i, y_i),\ i = 1,\cdots,n\}$, we create a new dataset by bootstrapping, i.e., randomly sampling the examples with replacement, and then grow a decision tree on this new dataset. For each split, instead of going over all the features, we use a random subset and only run through this subset of variables to decide on which variable we split. We then repeat this procedure B times to grow a large number of trees, i.e., a forest. The number of variables randomly sampled as candidates at each split is an important hyper-parameter that need to be tuned carefully during the model training stage. In the case of classification, the final prediction of random forest is based on the majority vote of classification results in B trees.

2.4 Gradient boosting

Gradient boosting is a machine learning technique for regression and classification problems. It combines weak prediction models into a single strong learner in an iterative fashion. Typically, shallow decision trees are used as weak learners in the gradient boosting model.

The gradient boosting algorithm is as follows in pseudocode:

Given $\{(x_i, y_i),\ i = 1,\cdots,n\}$ as training dataset, a differentiable loss function $L(y, F(x))$, and number of iterations M. In the case of binary classification, the loss function is defined as

$$L = -\frac{1}{n}\sum_{i=1}^{n}(y_i log(p_i) + (1 - y_i)log(1 - p_i)) \qquad (2.6)$$

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma}\sum_{i=1}^{n} L(y_i, \lambda) \qquad (2.7)$$

2. For m=1 to M:

A. Compute so-called pseudo-residuals:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \qquad for\ i = 1,\cdots,n \tag{2.8}$$

B. Fit a base learner $h_m(x)$ to pseudo-residuals, i.e., train it using the training set $\{(x_i, r_{im}),\ i = 1,\cdots,n\}$.

C. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i,\ F_{m-1}(x_i) + \gamma h_m(x_i)) \tag{2.9}$$

D. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \tag{2.10}$$

3. Output $F_M(x)$

In addition, for each gradient step, the step magnitude γ_m can also be multiplied by a factor between 0 and 1 called learning rate. Although gradient boosting model has many hyper-parameters that need to be tuned during the training stage, the most important ones are the maximum tree depth, number of iterations, and learning rate.

CHAPTER 3. Missing Value Problem And Imputation Algorithms

3.1 Nature of the problem

In statistics, missing values occur when no data value is stored for a particular variable in an observation. Missing data are a common problem and can have a significant effect on the conclusions that can be drawn from the data. By far, the most common means of dealing with missing data is to only include complete cases in the analysis, i.e, only keep those instances that have no missing data in any of the variables required for the analysis. However, results of such analyses can be biased if the observations with missing values differ systematically from the completely observed cases. Furthermore, the cumulative effect of missing data in several variables often leads to exclusion of a substantial proportion of the original sample, which in turn causes a substantial loss of information.

The risk of bias due to missing data depends on the reasons why data are missing. Reasons for missing data are commonly classified as: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR) [3].

- Missing completely at random - There are no systematic differences between the missing values and the observed values. For example, blood pressure measurements may be missing because of breakdown of an automatic sphygmomanometer.
- Missing at random - Any systematic difference between the missing values and the observed values can be explained by differences in observed data. For example, missing

blood pressure measurements may be lower than measured blood pressures but only because younger people may be more likely to have missing blood pressure measurements.

- Missing not at random - Even after the observed data are taken into account, systematic differences remain between the missing values and the observed values. For example, people with high blood pressure may be more likely to miss clinic appointments because they have headaches.

3.2 Missing value imputation

If data are missing completely at random, then throwing out examples with missing data will not bias the results and inferences. However, analyses only based on complete cases may be biased when the data are missing at random, but not completely at random. Such biases can be overcome by conducting missing value imputations. Many imputation approaches have been studied and reported by researchers. These include replacing missing values with the mean of the observed values for that variable, replacing missing values with the last measured value, and using information from related observations. However, none of these approaches is statistically valid in general, and they can lead to serious bias.

Another popular approach is to use indicator variables for missingness of categorical or continuous data. In particular, for categorical predictors, we can add an extra category for the variable indicating missingness. And for continuous predictors, the method is to include for each continuous predictor variable with missing values an extra indicator identifying which observations on that variable have missing data. Then the missing values in the partially observed predictor are replaced by mean. This strategy is usually appropriate for categorical data, but not very useful in the case of continuous data since missing values are still imputed by the mean, which in turn may cause serious bias on the results. Therefore, we need a more complex and subtle approach to impute missing values for the continuous data. Below we introduce two popular and principled methods for missing value imputation. Both of these two approaches can handle multivariate imputations on mixed-data type.

3.2.1 Multivariate imputation by chained equations

Multivariate imputation by chained equations (MICE) operates under the assumption that missing data are missing at random. In the MICE procedure a series of models are run whereby each variable with missing data is modeled conditional upon the other variables in the data. Specifically, linear regression and predictive mean matching is used to predict continuous missing values, logistic regression is used for binary variable missing values, bayesian multinomial logistic regression is used for categorical variables with more than 2 levels, and proportional odds model is used for ordered categorical variables with more than 2 levels.

The MICE steps are as follows [4]:

1. A simple imputation, such as imputing the mean, is performed for every missing value in the dataset. These mean imputations can be thought of as "place holders."
2. The "place holder" mean imputations for one variable ("var") are set back to missing.
3. The observed values from the variable "var" in step 2 are regressed on the other variables in the imputation model, which may or may not consist of all of the variables in the dataset.
4. The missing values for "var" are then replaced with predictions from the regression model. When "var" is subsequently used as an independent variable in the regression models for other variables, both the observed and these imputed values will be used.

5. Steps 2 through 4 are then repeated for each variable that has missing data. The cycling through each of the variables constitutes one iteration or "cycle." At the end of one cycle all of the missing values have been replaced with predictions from regressions that reflect the relationships observed in the data.
6. Steps 2 through 4 are repeated for a number of cycles, with the imputations being updated at each cycle. By the end of the cycles the distribution of the parameters governing the imputations (e.g., the coefficients in the regression models) should have converged in the sense of becoming stable.

The final imputations are retained at the end of these cycles, resulting in one imputed dataset.

3.2.2 MissForest

As the name suggests, MissForest is an implementation of random forest algorithm. It is able to deal with mixed-type data and as a non-parametric method it allows for interactive and non-linear effects. This method address the missing data problem using an iterative imputation scheme by training a random forest on observed values in a first step, followed by predicting the missing values and then proceeding iteratively [5].

To be more specific, we assume $X = (X_1, X_2, \cdots, X_p)$ is a $n \times p$ data matrix. Then for an arbitrary variable X_s including missing values at entries $i_{mis}^{(s)} \subseteq \{1, \cdots, n\}$, we can separate the dataset into four parts: The missing values of variable X_s, denoted by $y_{mis}^{(s)}$; The observed values of variable X_s, denoted by $y_{obs}^{(s)}$; The variables other than X_s with observations $i_{mis}^{(s)}$, denoted by $x_{mis}^{(s)}$; And the variables other than X_s with observations $i_{obs}^{(s)}$, denoted by $x_{obs}^{(s)}$. The algorithm begins with an initial guess for all missing values in X using mean imputation. Then, sort the variables

X_s, $s = 1,\cdots,p$ according to the amount of missing values starting with the lowest amount. For each variable X_s, the missing values are imputed by first fitting a random forest with response $y^{(s)}_{obs}$ and predictors $x^{(s)}_{obs}$, then predicting the missing values $y^{(s)}_{mis}$ by applying the trained random forest to $x^{(s)}_{mis}$. This imputation procedure is repeated until a stopping criterion is met.

CHAPTER 4. Class Imbalance Problem

4.1 Nature of the problem

Class imbalance problem typically refers to a classification problem where the classes are not represented equally. Generally speaking, any dataset can be considered as imbalanced dataset if the number of observations between classes are not equal. However, the common understanding of imbalanced dataset is that a dataset exhibits significant, and even extreme imbalanced [6]. As discussed in Chapter 1, learning from datasets with imbalanced class distribution is a difficult task since high accuracy can simply be achieved by just predicting all instances as the majority class in the imbalanced class scenario. The imbalanced nature of the data is typical of many applications such as medical diagnosis, credit card fraud detection, and text classification.

4.2 Reported research solutions

A number of solutions to the class imbalance problem are reported in the literature. These include the solutions at data level, algorithmic level, and cost-sensitive approach. The objective for methods at the data level is to rebalance the class distribution by resampling the data space. On the other hand, algorithmic-level approaches try to adapt existing classification algorithms, such as decision tree, to strengthen learning with regard to the minority class. Cost-sensitive learning is a type of machine learning that takes the misclassification costs into consideration. It assumes higher misclassification costs with samples in the minority class and the goal of this type of learning is to minimize the total cost.

4.2.1 Data-level approaches

Methods at data level include different kinds of resampling, such as under-sampling, over-sampling, and SMOTE (Synthetic Minority Over-sampling Technique). Under-sampling consists in removing examples in the majority class at random until the dataset is balanced, while over-sampling randomly replicates examples in the majority class.

SMOTE is another powerful over-sampling approach that was proposed by Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall and W. Philip Kegelmeyer in 2002. SMOTE was originally designed to conquer imbalanced datasets with numerical variables only. Given a dataset with imbalanced class distribution, the minority class is oversampled by taking each sample in the the minority class and creating N synthetic samples along with the K minority class nearest neighbors [6]. To be more specific, synthetic data points are generated in the following way: Take a minority class sample from the imbalanced dataset and compute its K nearest neighbors, then take the vector between the current data point and one of those K neighbors, multiply this vector by a random number between 0 and 1, and add it to the data point under consideration. As for generating synthetic samples for categorical variables, one simple and straightforward method is to randomly pick one neighbor from its K nearest neighbors and just assign the categorical feature value of that neighbor to the new synthetic sample. On the other hand, majority vote is another strategy for that implement more diversity into the algorithm. A couple of hyper-parameters need to be predefined in the algorithm: the number of synthetic samples we want to generate per original minority instance, and the number of neighbors we want to consider during the generation of each synthetic sample.

Although resampling is a widely used method in dealing with the class imbalance problem, there are known disadvantages associated with this method. The disadvantage with under-sampling is that it discards potentially useful data. The main disadvantage with over-sampling is that by making exact copies of existing examples, it makes overfitting likely [7]. Even for SMOTE, the performance is still extremely dependent on the data nature and distribution. For instance, adding too many similar synthetic data points will also lead to the increase of the probability of overfitting.

Another issue regards how to decide the optimal class distribution given a data set. A thorough experimental study on the effect of a training set's class distribution on a classifier's performance is conducted and the general conclusion is that, with respect to the classification performance on the small class, a balanced class distribution with approximate 1:1 class size ratio performs relatively well but is not necessarily optimal [8].

4.2.2 Cost-sensitive learning

Cost-sensitive classification takes the costs of different misclassification errors into consideration during model building. These costs can be given in a cost matrix, as shown in Table 4.1. In the table, $C(1,0)$ denotes the cost of false positive, $C(0,1)$ denotes the cost of false negative, $C(0,0)$ and $C(1,1)$ are the costs of true negative and true positive respectively. In the case of class imbalance problem, the minority class is usually regarded as the positive class, and the cost of false negative is usually more expensive than that of false positive. In addition, the costs of true negative and true positive are usually regarded as "benefit" and set as 0. That is, making a correct classification usually presents 0 penalty [8,9].

Table 4.1 Cost Matrix For Binary Classification

	Actual negative	Actual positive
Predict negative	C(0,0)	C(0,1)
Predict positive	C(1,0)	C(1,1)

Generally speaking, cost-sensitive learning techniques fall into two main categories. The first one is to design a specific cost-sensitive learning algorithm and directly utilize misclassification costs in the algorithm. Examples of this strategy include the cost-sensitive decision tree (Drummond and Holte, 2000; Ling et al, 2004). The cost information is used to choose the best split point and determine whether a subtree should be pruned.

The other category is to design a "wrapper" that converts an existing cost-insensitive classification algorithm into a cost-sensitive one. This can be easily accomplished by assigning different weights to positive and negative instances. According to Ref. [10], given the cost matrix and original threshold t_0 as 0.5, we keep the weight for negative class as 1, then the weight for observations in the positive class should be

$$W_{pos} = \frac{C(0,1) - C(1,1)}{C(1,0) - C(0,0)} \tag{4.1}$$

In addition, since we usually assume there is no penalty for correct classification, the equation above can be further simplified as

$$W_{pos} = \frac{C(0,1)}{C(1,0)} \tag{4.2}$$

That is, the weight for positive class should be equal to the cost of false negative divided by the cost of false positive. In this paper, we use the wrapper-based approach (second one) to build cost-sensitive classification models on the hospital readmission dataset.

4.2.3 Neyman-Pearson classification

Neyman-Pearson (NP) classification paradigm is designed to address the need to limit the more severe type of error so that it remains below a desired threshold. It provides the ability of probabilistic control on the errors by minimizing the less severe type of error while enforcing an upper bound, α, on the more severe type of error. A main advantage of the NP classification is that it is a general framework that allows users to find classifiers with the more severe type of error under α with high probability. However, it can not guarantee that the less severe type of error is small or the overall classification error is small. To reduce the other two errors, the modification should be made to the classification method, e.g., replacing the logistic regression by Neural Network. Although the NP paradigm has a long history in hypothesis testing, it has not been well recognized and implemented in classification schemes, primarily because of the challenge on how to implement it with diverse classification algorithms. In Ref. [12], the authors proposed an umbrella algorithm to implement a broad class of classification methods under the NP paradigm, along with a visualization tool for NP classification - NP receiver operating characteristic (NP-ROC) bands. The NP-ROC bands can be used to choose α in a data-adaptive way and compare different NP classifiers. It is important to note that the Neyman-Pearson classification is not a competitor of various classification methods, but an umbrella algorithm that is compatible with those methods. Therefore, the NP umbrella algorithm and NP-ROC bands together provide a flexible pipeline to implement binary classification in broad applications where controlling the prioritized type of error is needed.

4.3 Evaluation metrics for class imbalance problem

Evaluation metric is a crucial measure for both the assessment of classification models and hyper-parameter tuning during the training stage. Traditionally, accuracy is the most common measure for classification problem

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.3}$$

where TP, FP, FN, TN are defined in the confusion matrix in Table 4.2. However, for class imbalance problem, accuracy is no longer a proper measure since the minority class has very little impact on accuracy as compared to the majority class.

Table 4.2 Confusion Matrix

	Predicted as positive	Predicted as negative
Actually positive	True positives (TP)	False negatives (FN)
Actually negative	False positives (FP)	True negatives (TN)

Other traditional evaluation metrics include F1-score, G-mean, and AUC-ROC (Area Under Receiver Operating Characteristic Curve). F1-score can be interpreted as the harmonic mean between precision and recall and is a proper measure when only the performance of the positive class prediction is considered.

$$Precision = \frac{TP}{TP + FP} \tag{4.4}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.5}$$

$$F1\ Score = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{4.6}$$

G-mean is a more appropriate evaluation metric when the performance of both classes is concerned. It is defined as the following

$$G - mean = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \tag{4.7}$$

The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings, where the definitions of TPR and FPR are listed as below.

$$TPR = \frac{TP}{TP + FN} \tag{4.8}$$

$$FPR = \frac{FP}{FP + TN} \tag{4.9}$$

The area under the ROC curve is a widely used evaluation metric in machine learning community for the purpose of model comparison, the higher the value is, the better classification performance the model has.

In the case of cost-sensitive learning where the cost matrix is given as Table 4.1, the best metric for evaluating classifier performance is total misclassification cost. The formula below shows how we calculate the total cost, where N_{fn} is the number of false negative predictions and N_{fp} is the number of false positive predictions.

$$Total\ Cost = N_{fn} \times C(0,1) + N_{fp} \times C(1,0) \tag{4.10}$$

CHAPTER 5. Experiment

5.1 Methodology

As mentioned in Chapter 1, the hospital readmission dataset we use in this paper contains 552,816 observations and 116 predictor variables, and there are two issues in the dataset we need to handle before building classification models: high percentage of missing values and imbalanced class distribution. Therefore, missing value imputations are first conducted on the dataset. For categorical variables with missing data, we add an extra category "unknown" in each variable and use it to replace missing values. For numerical variables with missing data, a second dummy variable is created per numeric to identify which observations on that variable have missing data. Then MICE is applied to impute missing values in those numerical variables. The binary response variable and newly created dummy variables are not included in the data matrix for MICE imputation.

Next we randomly split the dataset into 3 parts, training set, validation set, and test set. The training set with 288,681 negative examples and 43,010 positive examples is used for model fitting. Hyper-parameter tuning and model comparison are conducted by using the validation set and test set respectively.

The main part in this experiment is to implement resampling method and cost-sensitive learning separately on our hospital readmission dataset and compare which approach will yield a better classification performance.

- For the resampling method, we assume the cost matrix is not known in advance. Therefore, according to our discussions in Section 4.2.1, SMOTE is first applied to create a new balanced training set (approximate 1:1 class size ratio) by generating synthetic data and downsizing the majority class simultaneously on the original training set. Then three different classification models including LASSO logistic regression, random forest, and gradient boosting machine are fitted on the rebalanced training data. During the training stage, hyper-parameters are tuned based on the AUC-ROC value as shown in Figures 5.1-5.3. Particularly, for each of the three classification methods, the hyper-parameter values that yield highest AUC on validation set are selected for the training of the final model.

- In the scenario of cost-sensitive learning, the misclassification costs are known in advance so that we can calculate the weights for positive and negative examples separately based on our discussions in Section 4.2.2. According to Ref. [11], the cost matrix for hospital readmission is shown in Table 5.1. Assuming a weight of 1 for observations in negative class, then the weight for examples in positive class is calculated by equation (4.2), which yields a value of 7.125. Then LASSO logistic regression, random forest, and gradient boosting machine are fitted on the original training data but with different weights on positive and negative instances. During the training stage, hyper-parameters are tuned based on the total misclassification cost as shown in Figures 5.4-5.6. Particularly, for each of the three classification methods, the hyper-parameter values that yield lowest misclassification cost on validation set are selected for the training of the final model.

Table 5.1 Cost Matrix For Hospital Readmission

	Actual negative	Actual positive
Predict negative	\$0	\$5700
Predict positive	\$800	\$0

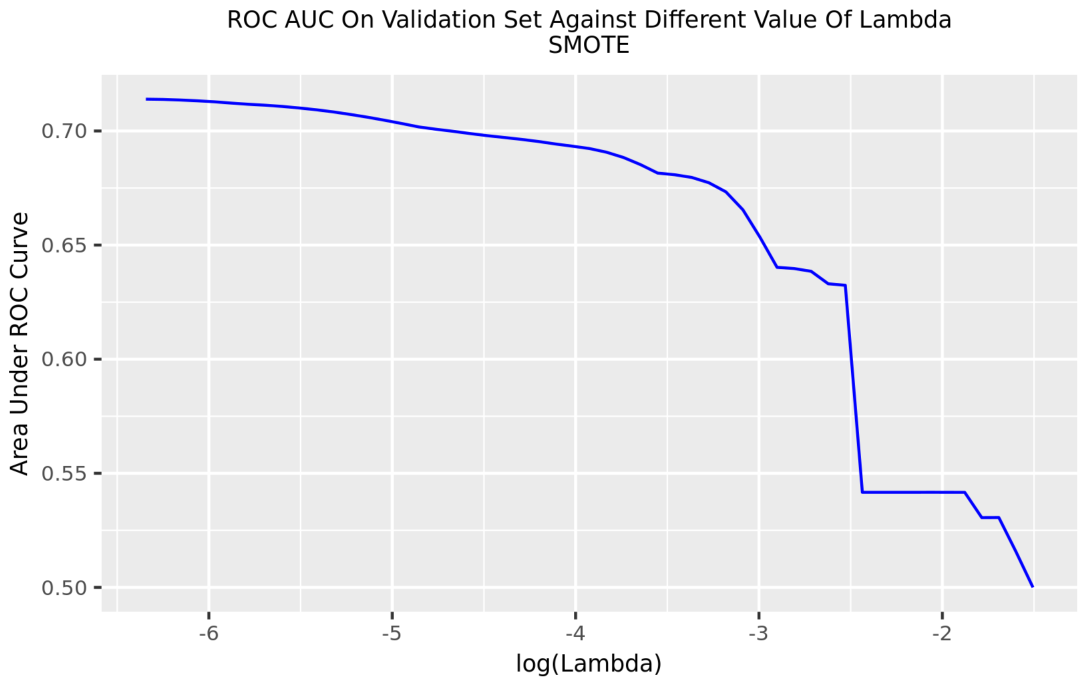

Figure 5.1 Hyper-parameter Tuning For LASSO Logistic Regression With SMOTE

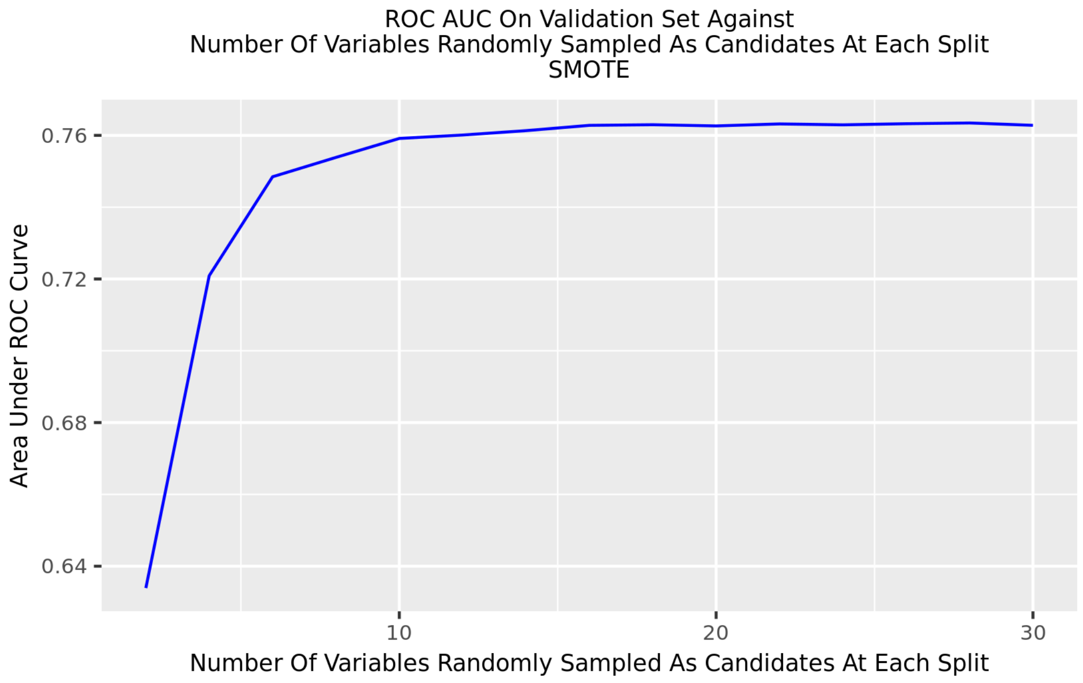

Figure 5.2 Hyper-parameter Tuning For Random Forest With SMOTE

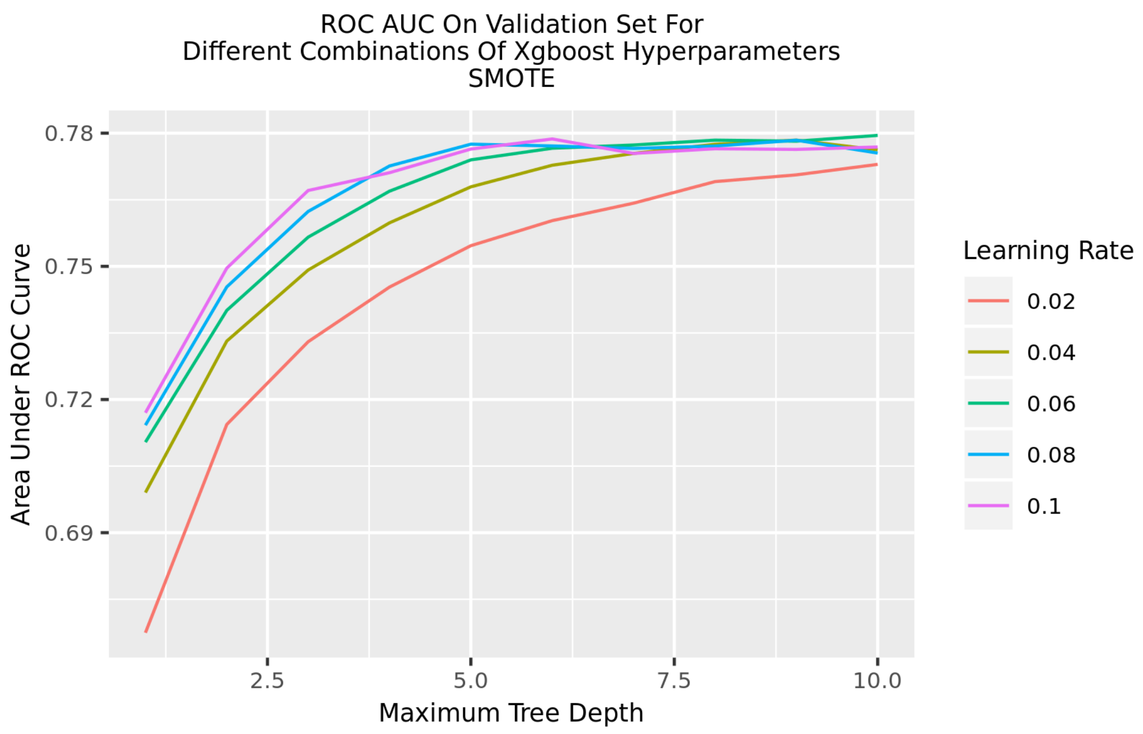

Figure 5.3 Hyper-parameter Tuning For Gradient Boosting With SMOTE

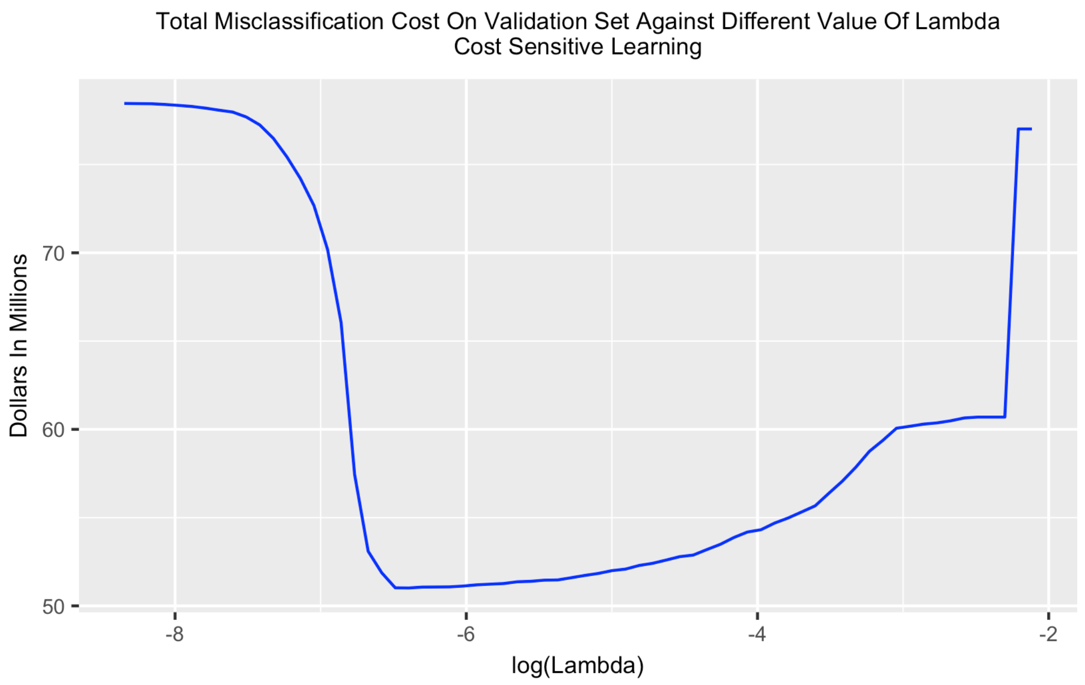

Figure 5.4 Hyper-parameter Tuning For LASSO Logistic Regression With Cost Sensitive Learning

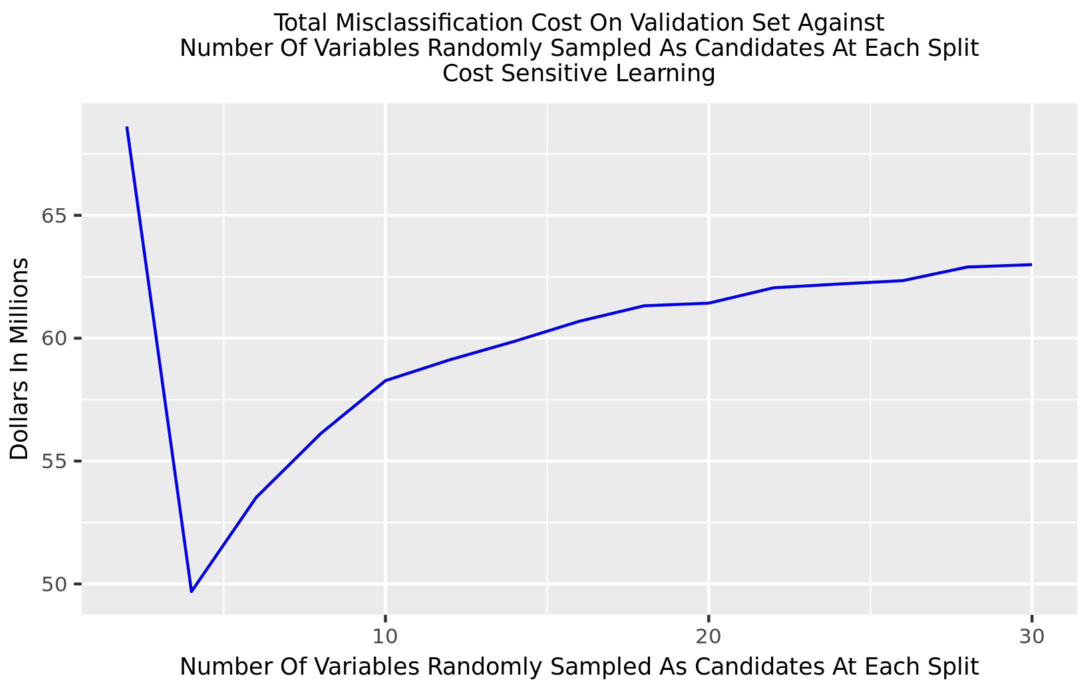

Figure 5.5 Hyper-parameter Tuning For Random Forest With Cost Sensitive Learning

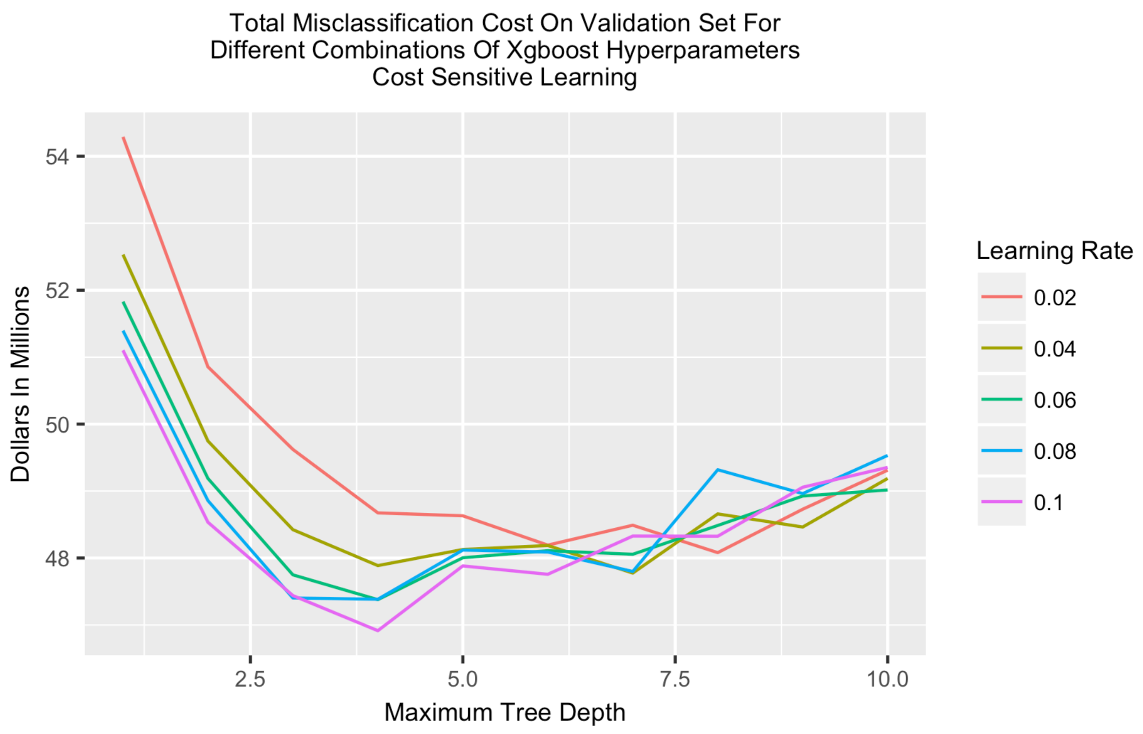

Figure 5.6 Hyper-parameter Tuning For Gradient Boosting With Cost Sensitive Learning

5.2 Results

By far, total of six classification models have been trained. These include LASSO logistic regression, random forest, and gradient boosting machine fitted on the SMOTEd data, and those three classification models trained by using the cost-sensitive learning approach. The performance of these six models are compared by using ROC curve and total misclassification cost on the test dataset. Results are exhibited in Figures 5.7-5.14.

In particular, the ROC curves on training and test sets for LASSO logistic regression, random forest, and gradient boosting machine fitted on the SMOTEd data are shown in Figures 5.7-5.9. Figures 5.10-5.12 plot ROC curves on training and test sets for models trained by using the cost-sensitive learning. The AUC-ROC value and total misclassification cost on test set for each of the models are plotted in Figure 5.13 and Figure 5.14 respectively.

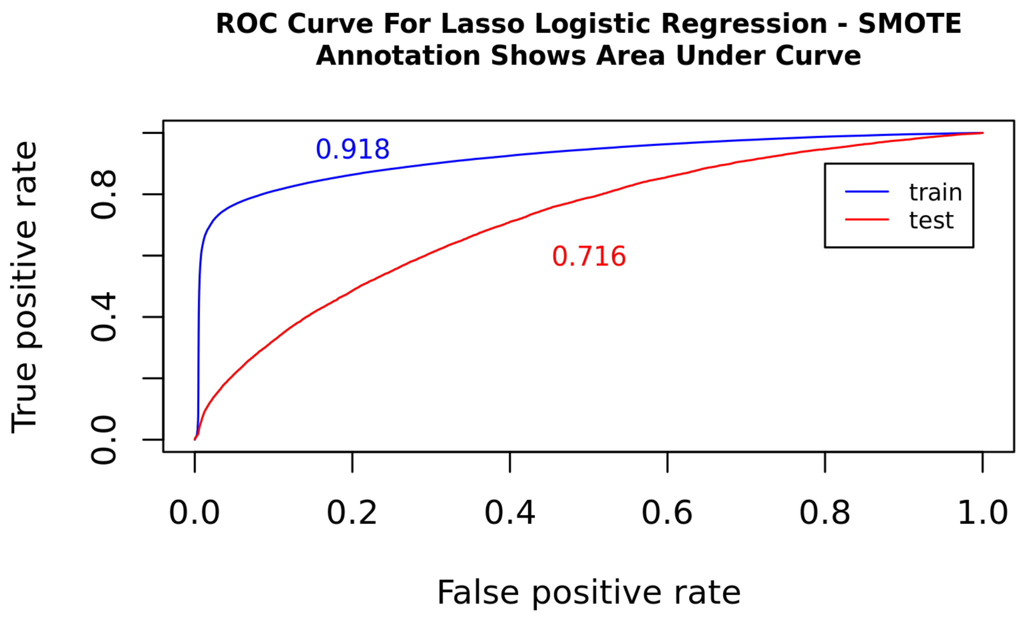

Figure 5.7 ROC Curve For Lasso Logistic Regression With SMOTE

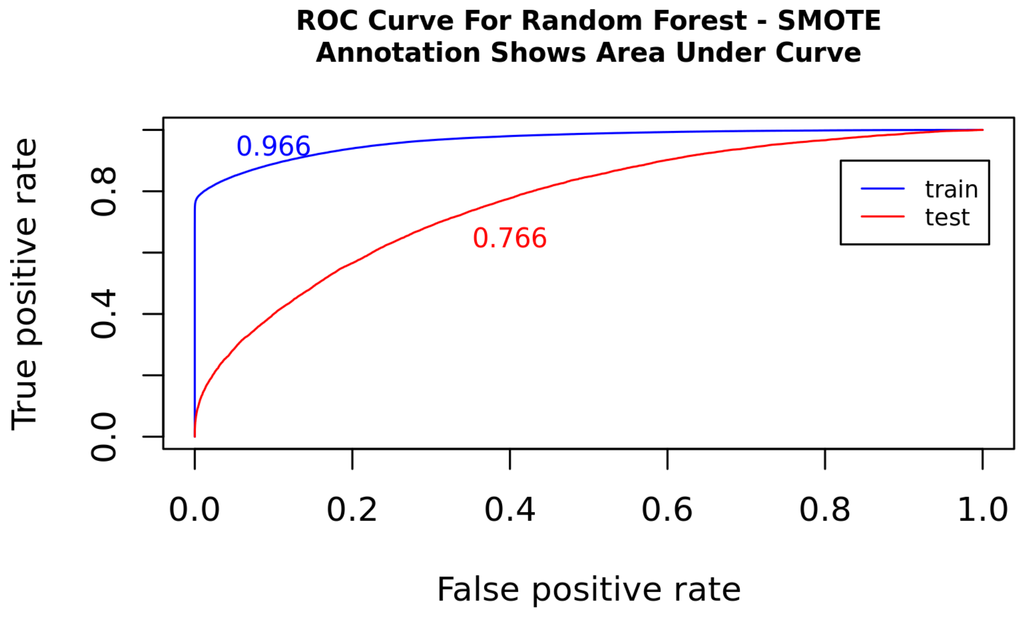

Figure 5.8 ROC Curve For Random Forest With SMOTE

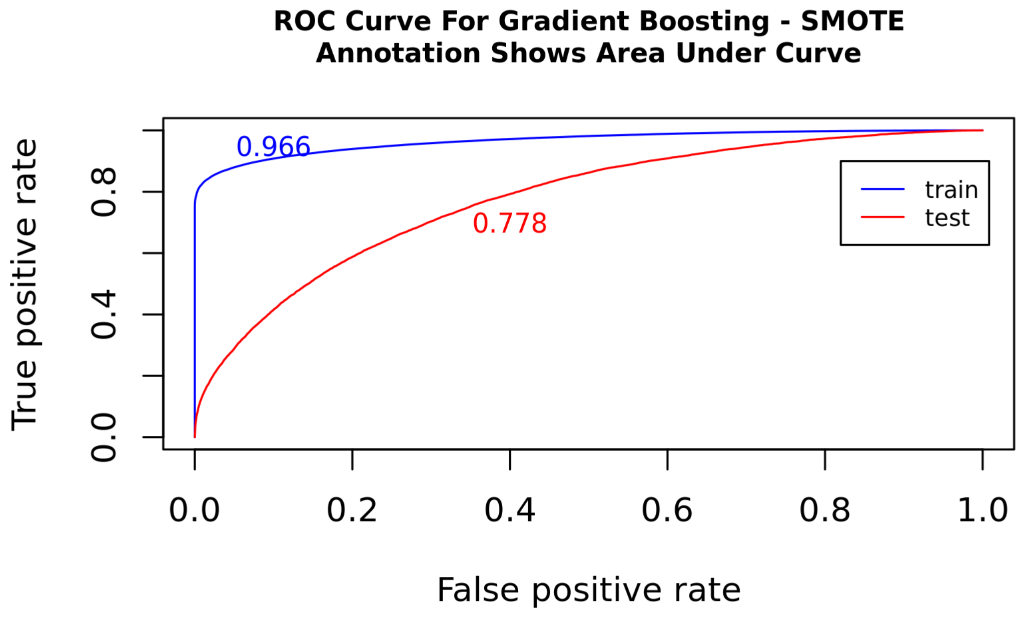

Figure 5.9 ROC Curve For Gradient Boosting With SMOTE

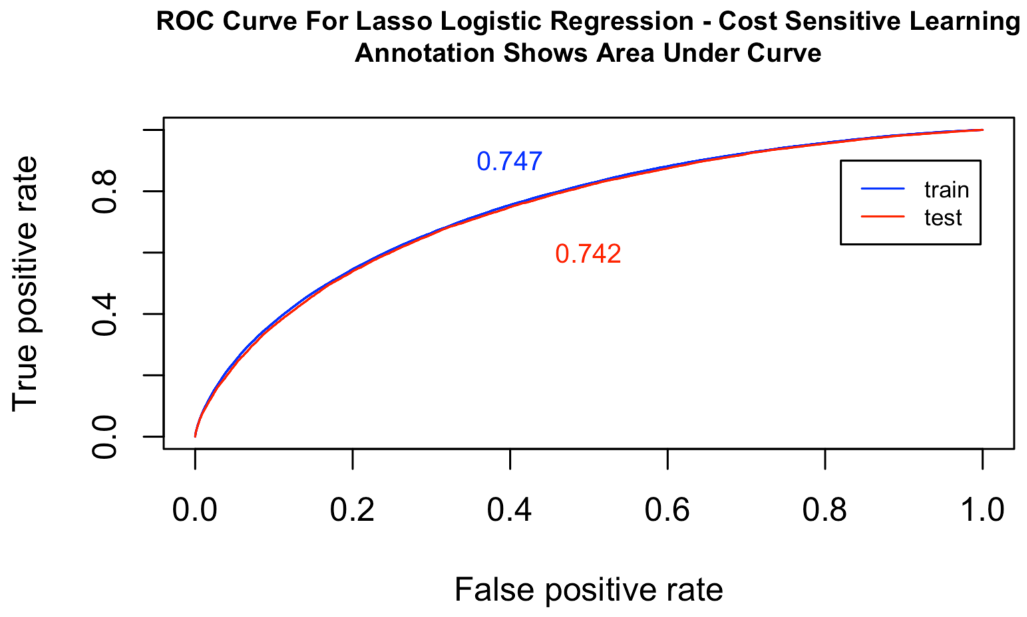

Figure 5.10 ROC Curve For Lasso Logistic Regression With Cost Sensitive Learning

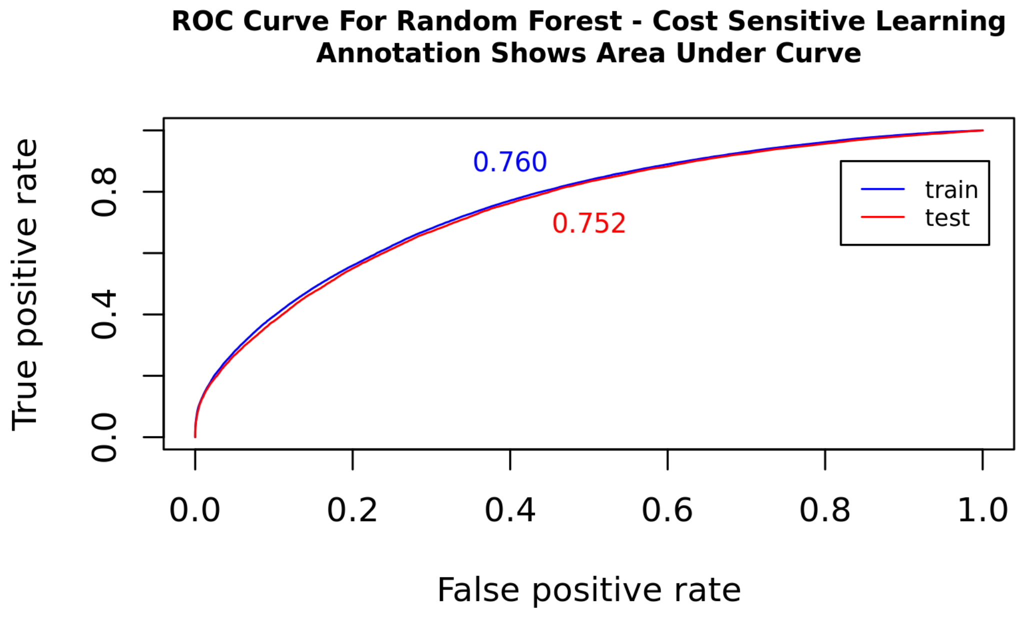

Figure 5.11 ROC Curve For Random Forest With Cost Sensitive Learning

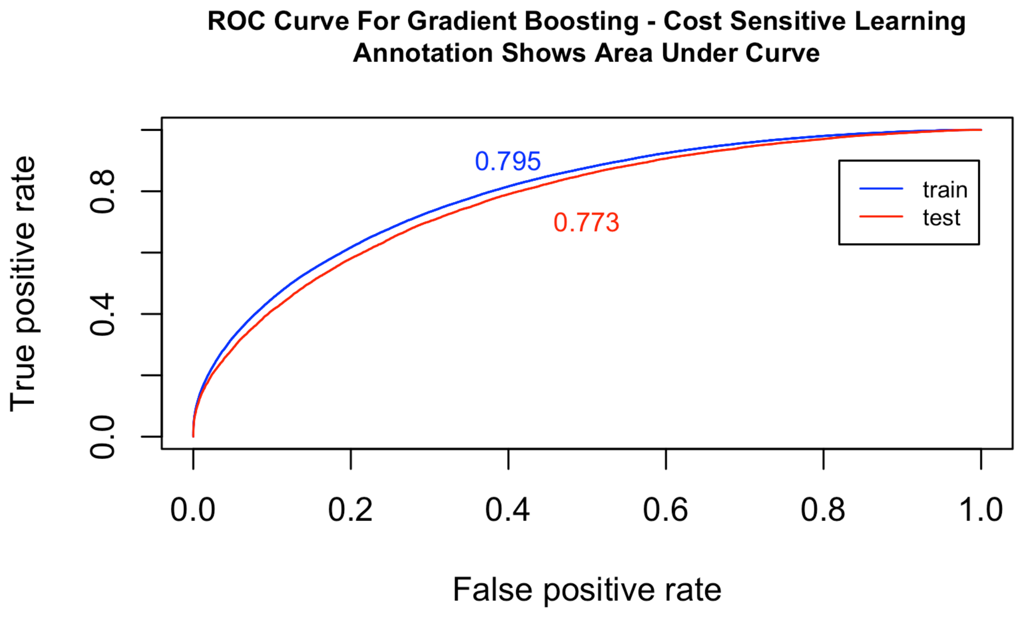

Figure 5.12 ROC Curve For Gradient Boosting With Cost Sensitive Learning

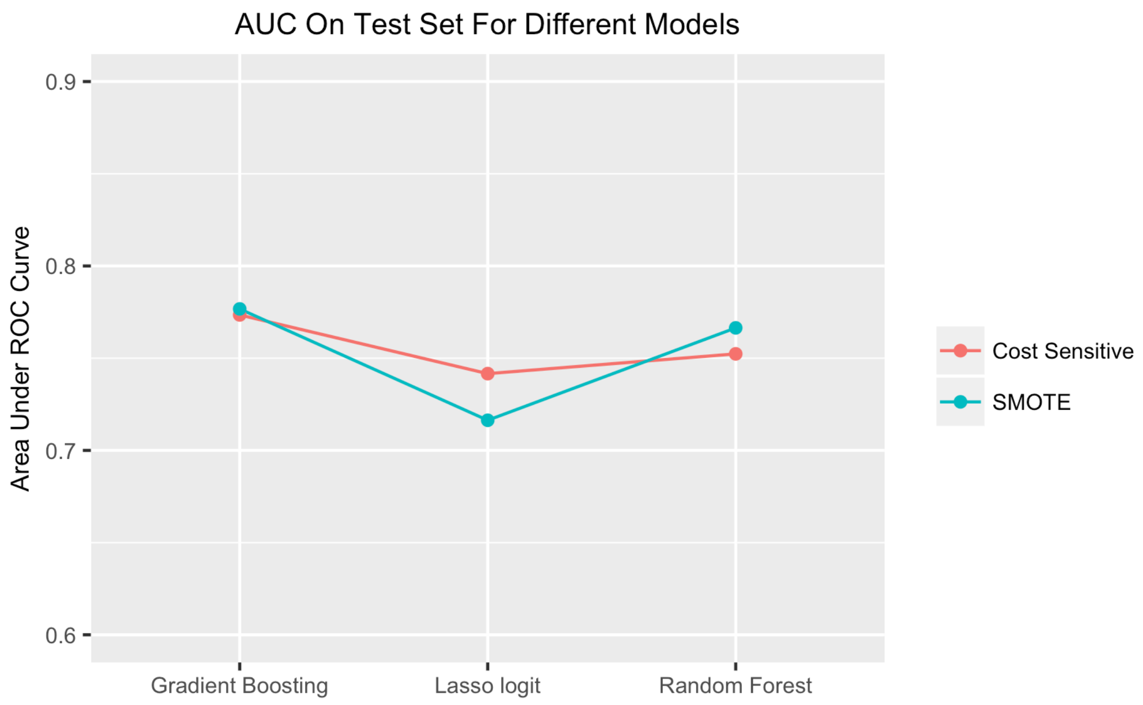

Figure 5.13 Model Comparison By AUC On Test Set

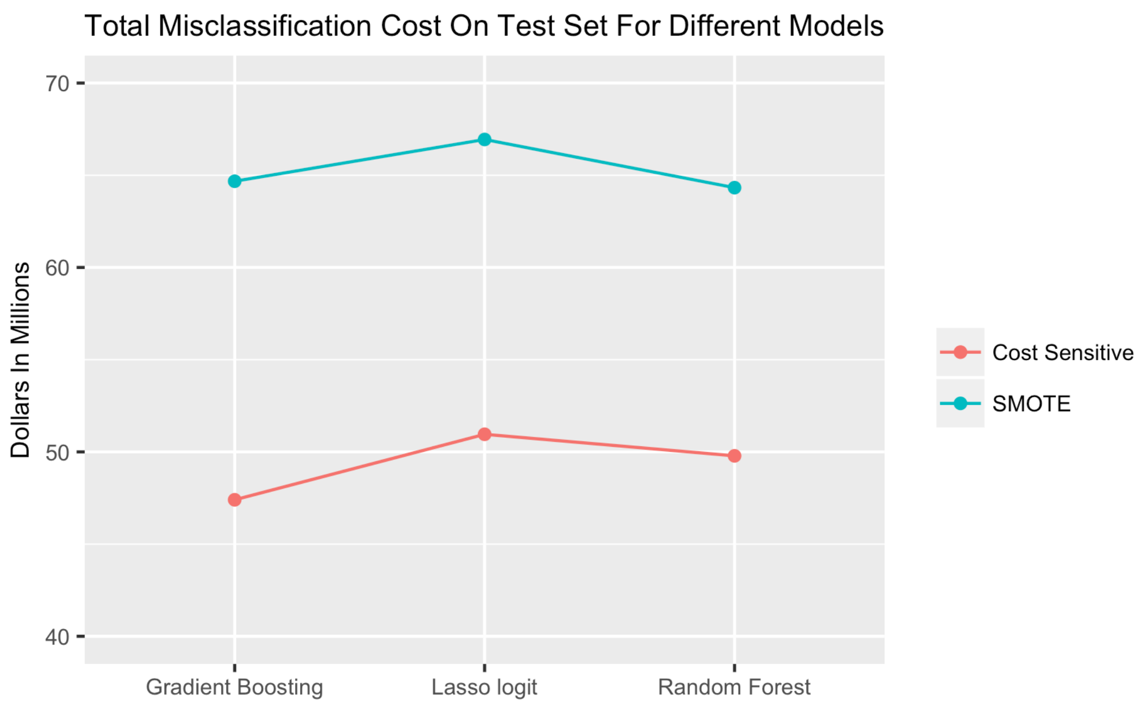

Figure 5.14 Model Comparison By Total Misclassification Cost On Test Set

CHAPTER 6. Conclusions

In this paper, six misclassification models are fitted on the hospital readmission data after missing value imputations. These models include LASSO logistic regression, random forest, and gradient boosting machine fitted on the SMOTEd data, and those three classification models trained by using the cost-sensitive learning approach. The performance of these six models are compared and displayed in Section 5.2. According to the results, we can draw conclusions as listed below.

1. The classification models fitted on the SMOTEd readmission data are overfitted on the training set. This is clearly shown in Figures 5.7-5.9 where the AUC values on training set are significantly higher than that on test set. This result supports the idea that over-sampling methods make overfitting more likely by making exact copies or even synthetic data of existing examples
2. Although there is no significant difference on the AUC value between models fitted on the SMOTEd data and models trained by using the cost-sensitive approach, cost-sensitive learning outperforms SMOTE with regard to total misclassification cost.
3. According to the Figure 5.14, the gradient boosting model with cost-sensitive learning approach returns the lowest misclassification cost on test set. This model also yields one of the highest AUC values on the test set.

References

[1] H. Wang, Z. Cui, Y. Chen, M. Avidan, A. B. Abdallah and A. Kronzer (2018). Predicting Hospital Readmission via Cost-sensitive Deep Learning. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*. doi:10.1109/TCBB.2018.2827029

[2] Dal Pozzolo, Andrea & Caelen, Olivier & Waterschoot, Serge & Bontempi, Gianluca (2013). Racing for Unbalanced Methods Selection. *Intelligent Data Engineering and Automated Learning - IDEAL 2013: 14th International Conference, IDEAL 2013, Hefei, China, October 20-23, 2013. Proceedings.* 8206. doi:10.1007/978-3-642-41278-3_4

[3] Sterne JAC, White IR, Carlin JB, et al. (2009). Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. *The BMJ.* 2009;338:b2393. doi: 10.1136/bmj.b2393.

[4] Azur MJ, Stuart EA, Frangakis C, Leaf PJ. (2011). Multiple Imputation by Chained Equations: What is it and how does it work? *International journal of methods in psychiatric research*. 2011;20(1):40-49. doi:10.1002/mpr.329.

[5] Daniel J. Stekhoven, Peter Bühlmann (2012). MissForest—non-parametric missing value imputation for mixed-type data, *Bioinformatics*, *Volume 28, Issue 1, 1 January 2012*, Pages 112–118. https://doi.org/10.1093/bioinformatics/btr597

[6] Gao, Tianxiang (2015). Hybrid classification approach for imbalanced datasets. *Graduate Theses and Dissertations.* 14331. https://lib.dr.iastate.edu/etd/14331

[7] Weiss, G.M., McCarthy, K., & Zabar, B. (2007). Cost-Sensitive Learning vs. Sampling: Which is Best for Handling Unbalanced Classes with Unequal Error Costs? *DMIN.*

[8] Yanmin Sun, Mohamed S. Kamel, Andrew K.C. Wong, Yang Wang (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition, Volume 40, Issue 12, December 2007*, Pages 3358-3378. https://doi.org/10.1016/j.patcog.2007.04.009

[9] N. Thai-Nghe, Z. Gantner and L. Schmidt-Thieme (2010). Cost-sensitive learning methods for imbalanced data. *The 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, 2010*, pp. 1-8. doi: 10.1109/IJCNN.2010.5596486

[10] Bernd Bischl, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter, Erich Studerus, Giuseppe Casalicchio, Zachary M. Jones (2016). mlr: Machine Learning in R. *Journal of Machine Learning Research.* 17(170):1-5, 2016.

[11] The Cost of a Hospital Readmission (2016). Retrieved from https://www.speechmed.com/cost-hospital-readmission

[12] Xin Tong, Yang Feng, Jingyi Jessica Li (2018). Neyman-Pearson classification algorithms and NP receiver operating characteristics. *Science Advances, Vol. 4, no. 2, February 2018*. https://doi.org/10.1126/sciadv.aao1659