Research on Neyman-Pearson (NP) Classification Algorithms

Jiayu Wu, Yifei Xu, Linfan Zhang
Department of Statistics
University of California, Los Angeles
5632 Geology Bldg., Los Angeles, California 90095
{jiayuwu, fei960922, linfanz}@ucla.edu

Abstract

Controlling type I error (i.e., the conditional probability of misclassifying a class 0 observation as class 1) in binary classification is a primary concern in cases where a) the training data is severely biased; b) the type I error incurs a higher cost in application, such as disease diagnosis and spam detection. Neyman-Peason(NP) lemma can be applied to address this issue. Inspired by the NP umbrella algorithm and the graphical tool NP-ROC developed by Xin Tong, el ta., we proposed a threshold selection algorithm using booststrap. And then we compared the performance of these two algorithms using simulated data. Finally, we applied NP classification and NP-ROC to a biased data set with dominated class 1 observations to evaluate their effeciency.

1 Introduction

Classification methods mainly focus on minimizing the overall risk, which can be viewed as the weighted sum of type I and type II error. In binary classification paradigm,Type I error is defined as the probability of asssigning a class 0 observation to class 1. Type II error, on the other hand, is defined as the probability of assigning a class 1 observation to the class 0. The corresponding weights are marginal probability of an observation being in class 0 or class 1. However, in real life application, we may be specifically interested in controlling type I error instead of controlling the risk as a whole. We can see many examples in medical classification problems. The consequences of misclassifying a patient as healthy are much more severe than missclassifying a healthy person as sick. In addition, the marginal probability of being a patient is relatvely bigger in training data. Therefore, we will give greater penalty on misclassifying a healthy person to the patient group, which is not appropriate considering our goal. A classification method focused on controlling type I error thus would be useful in solving this problem.

A first approach to be considered is to apply Neyman-Pearson lemma of hypothesis testing in scoring type of classification methods. In this report, we demonstrate three methods to select a threshold, refered to as naive NP classifier, boostrap NP classifier and order NP classifier. Based on simulation results, it is proven that NP classifier developed by Prof. Jessica Li outperformed others, then application on real data is discussed to demonstrate the advantages of this approach.

2 Backgrounds

Let $\phi : \mathbb{R}^d \to \{0, 1\}$ be a classifer on random variable $\{X, Y\}$, where $X \in \mathbb{R}^d$, $Y \in \{0, 1\}$. Then the population type I error is defined as $R_0(\phi) = \mathbb{P}(\phi(X) = 1|Y = 0)$, and the population type II error is defined as $R_1(\phi) = \mathbb{P}(\phi(X) = 0|Y = 1)$. Instead of finding a so-called *classical oracle*

classifier that minimizes the risk,

$$\phi^* = \arg\min_{\phi} R(\phi) = \arg\min_{\phi} R_0(\phi)\mathbb{P}(Y=0) + R_1(\phi)\mathbb{P}(Y=1)$$

NP classification aims to finding *NP oracle classifier* that minimizes Type II error when bounding Type I error smaller than a preset value α, which is very similar to the Neyman-Pearson lemma:

$$\phi_\alpha^* = \underset{\phi: R_0(\phi)] \leq \alpha}{\arg\min}\ R_1(\phi)$$

In the scoring type of classification, we can acheive this goal by selecting a proper threshold. To be specific, we first apply the classification method on our training set to get the score function $f(x)$, then a classifier would be determined by setting a threshold θ to this score function, i.e. $\phi(x) = I\{f(x) > \theta\}$. The threshold is selected based on the NP criterion stated above.

The major concern is that since we have no clue of the true underlying distribution of random variable $\{X, Y\}$, R_0 and R_1 are unobservable. The most straightforward approach is to use training and testing datasets, by applying the trained classification model to the test set, the empirical Type I and Type II error can be calculated, and select the threshold accordingly. However, we will demonstrate that this naive way would fail to control the population Type I error with high probability. We then came up with estimating Type I and Type II error by boostrap, which would be a great improvement from the naive NP classification. However, considering the computation cost and the uncertainty of controlling type I error, we found that NP classification developped by order statistics are more efficient ([1]). After comparing those three ways to mimic oracle NP classification, we apply the order NP classification to the real data to check its performance.

3 Naive NP classification

Naive NP classification mimics oracle NP classifier by controling empirical type I error and selecting the minimal threshold that has empirical type I error smaller that the preset α level to maximize power(1 - type II error). Denote the number of samples in the training set as n_0, Naive NP classification select the threshold by:

$$\theta = \inf\left\{ \frac{\sum_{i=1}^{n_t} I\{f(x_i) > \theta, y_i = 0\}}{\sum_{i=1}^{n_t} I\{y_i = 0\}} \right\}$$

Using empirical type I error to represent population type I error can be problematic because the threshold derived in this way can not bound population type I error with high probability (see the simulation part). The main reason is that we only get one realized value for poulation type I error, which is a random variable in essence. This property also restricts our ability to control type I error tightly, so the following two algorithms to select the threshold can only ensure that the population type I error is smaller than the preset α with high probability $1 - \delta$. Here, δ is also refered to as the violation rate in the following context. It can be interpreted as the probability of the type I error greater than the bound α, so it is a function of α.

4 Bootstrap NP classification

A natural idea to approximate the distribution of type I error would be bootstrap. For each candidate threshold calculated based on the score funciton of training set, we can bootstrap over the testing set B times to get the distribution of the empirical type I error, which is an approximation to the underlying true distribution of population type I error. Then we can calculate the violation rate for each threshold:

$$V(\theta) = \frac{1}{B}\left(\#\text{of emprical type I error} > \alpha\right)$$

and select the minimum threshold whose violation rate $R(\theta)$ is smaller than δ. To sum up, boostrap NP classification select the threshold by

$$\theta = \inf\{R(\theta) < \delta\}$$

4.1 Algorithms

We will provide three equivalent algorithms for bootstrap. The first one is the most straightforward. With trained model, a threshold and a expected type-I error, bootstrap can be used to estimate the "violation rate", which is the percentage of bootstrap round which suits the required type-I error, which is, whether the type-I error of predicting the boostrapped data by trained model and the threshold exceedes the expected one. For a fixed model, different threshold led to different "violation rate". Our algorithm will output the highest threshold with biggest "violation rate" smaller than our expectation.

The implementation detail of this method is listed in algorithm 1. Notice that in order to get the biggest one smaller than expected value, a for-loop decreasing from big to small can be employed, and the first one suits the requirement is returned.

Algorithm 1 Bootstrap Naive

procedure ESTIMATE(THRESHOLD : θ, SCORE FUNCTION : $f(x)$, EXPECTED ERROR : α)
    $numSatisfied \leftarrow 0$
    for B times **do**
        $X_b \leftarrow$ random sample n data from X (with replacement)
        $error \leftarrow$ Apply $f(x)$ and threshold θ to predict the data X_b, record the type-I error.
        if $error < \alpha$ **then** $numSatisfied \leftarrow numSatisfied + 1$
    $Violationrate \leftarrow numSatisfied/B$
    return Violation rate
procedure BSCLASSIFIER(EXPECTED ERROR : α, EXPECTED VIOLATION RATEδ)
    Train the original classifier, Let it as f(X), which is a score scale from 0 to 1
    for threshold θ = 0 to 1 **do**
        $vRate_\theta \leftarrow estimate(f(x), \theta, \alpha)$.
        if $vRate_\theta < \delta$ **then return** θ as threshold

It is obvious that as the threshold increases, the violation rate also increases. So we can use dichotomy to improve the efficiency.

Algorithm 2 Bootstrap Dichotomy

procedure BSCLASSIFIER(EXPECTED ERROR : α, EXPECTED VIOLATION RATEδ)
    Train the original classifier, Let it as f(X), which is a score scale from 0 to 1
    $threshold \leftarrow 0.5$
    $l \leftarrow 0; r \leftarrow 1$
    while True **do**
        $vRate_\theta \leftarrow estimate(f(x), \theta, \alpha)$.
        if $vRate_\theta < \delta$ **then** $\theta = (r + \theta)/2$
        else$\theta = (l + \theta)/2$
        if $vRate_\theta == \delta$ **then** Break

There is a third algorithm to this end, where we put the for-loop inside the bootstrap. For each bootstrap epoch, we find the thresholds which correspond to the expected type-I error. Note that here the type-I error is empirical. We still need multiple rounds of bootstrap to estimate the population one. Rather than predict the violation rate for every threshold, we transform the violation rate to threshold ranking.

Each bootstrap outputs a threshold. By sorting them in order, the k-th threshold where $k = n * \delta$ is chosen. For example, $\delta = 0.1, n = 500$, we choose 50th biggest threshold as the final output.

The algorithm is as the following:

Algorithm 3 Bootstrap the threshold

procedure BSCLASSIFIER-33(EXPECTED ERROR : α, EXPECTED VIOLATION RATE:δ)
  Train the original classifier, Let it as f(X), which is a score scale from 0 to 1
  for B times **do**
    $X_b \leftarrow$ random sample n data from X (with replacement)
    Apply $f(x)$ to predict the data X_b
    if $error < \alpha$ **then** $numSatisfied \leftarrow numSatisfied + 1$
  $threshold \leftarrow 0.5$
  $l \leftarrow 0; r \leftarrow 1$
  while True **do**
    $vRate_\theta \leftarrow estimate(f(x), \theta, \alpha)$.
    if $vRate_\theta < \delta$ **then** $\theta = (r + \theta)/2$
    else$\theta = (l + \theta)/2$
    if $vRate_\theta == \delta$ **then** Break

It is not surprising that the above three methods are equivalent.

4.2 Visualization

In this section, we perform simulation on random sampled data to validate the previous reasoning. The sampled dataset is the same as the previous. The violation rate is 0.067, which is very close to the result of another equivalent algorithm.

Visualization of our results is presented as follows. For algorithm 1, we plot the the histgram of empirical error in multiple bootstarp for a particular threshold. We can see that with a large number of epochs, the histgram tends to normal distribution with a mean at around the population error.

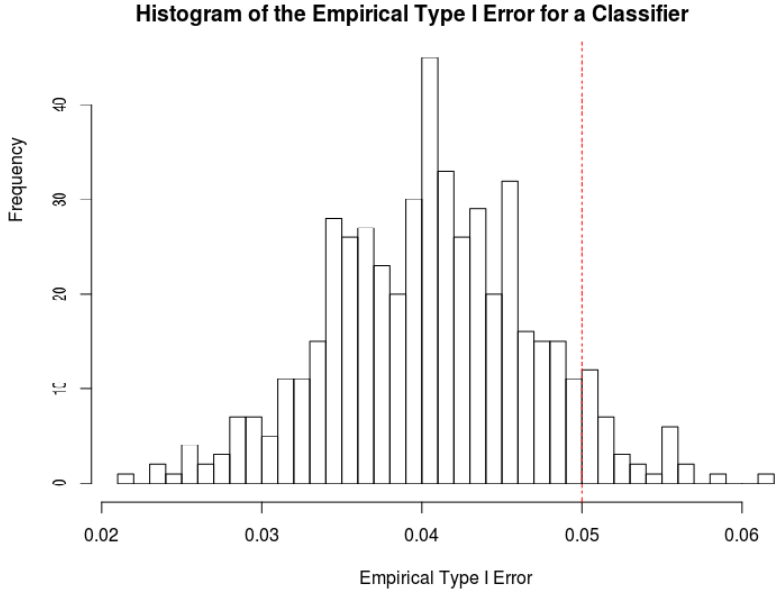


Figure 1: Histgram of empirical error for a peticular threshold

Then we plot histgrams corresponding to 10 different thresholds. We can see that the bigger threshold lead the bigger mean of empirical error, while the variance are similar.

The distribution of bootstrapped violation rate with threshold is presented in the figure below. Actually the violation rate is the area on the left side of $x = \alpha$ on the previous histgram. It can be observed that bigger threshold led to the bigger violation rate.

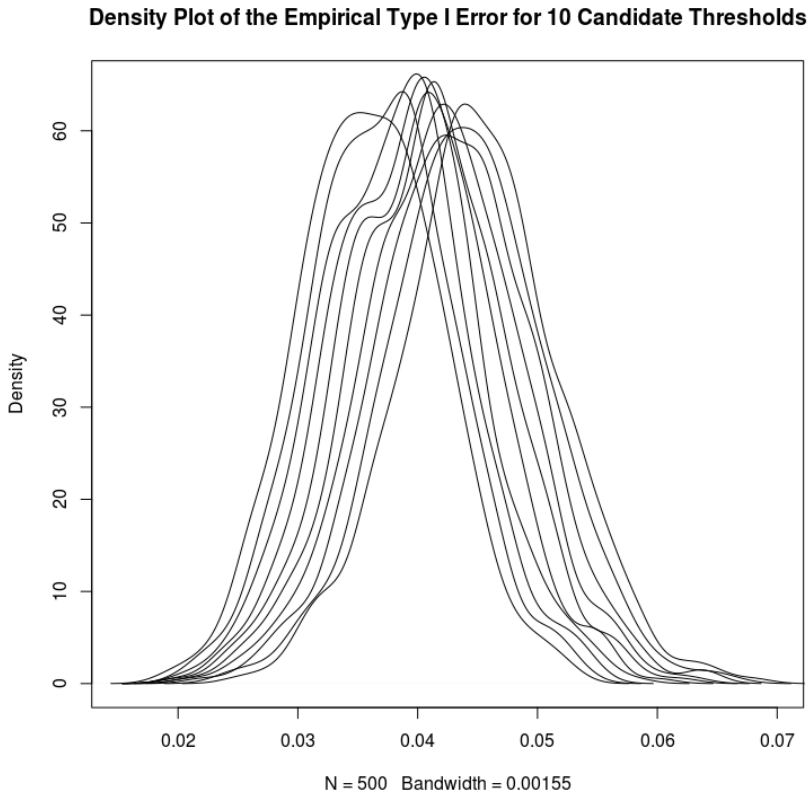
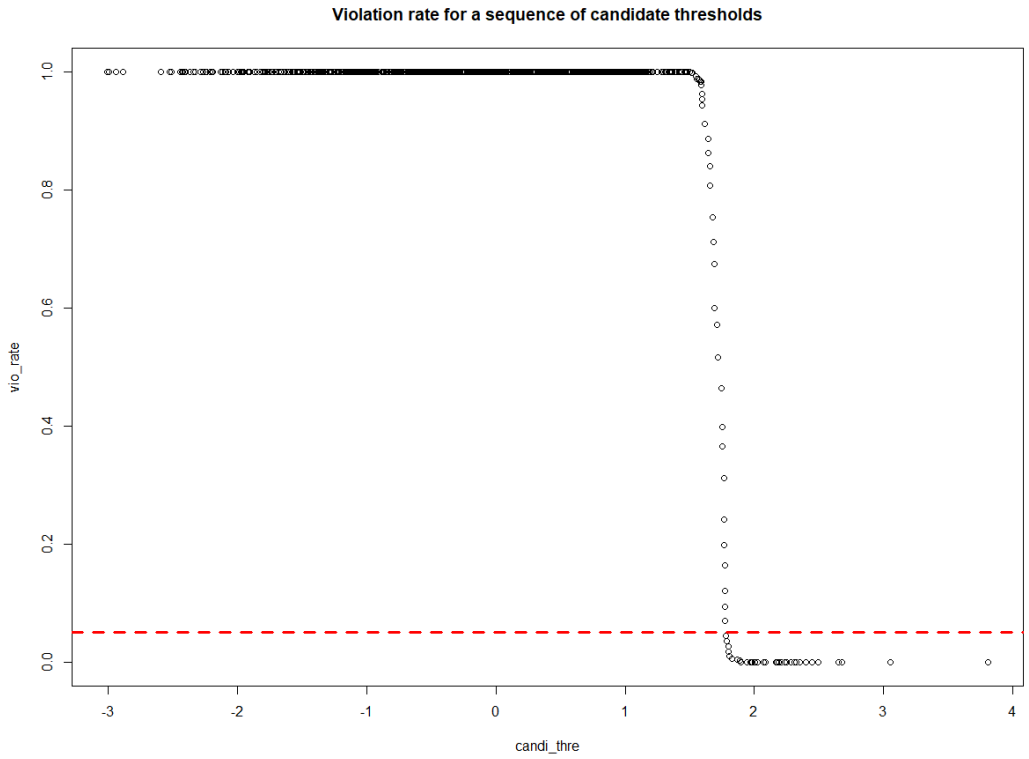


Figure 2: Histgram of empirical error for 10 different threshold

Then we plot the threshold-estimated-population-error. This is CDF of the previous histgram, i.e., $vRate$ in algorithm 1. For algorithm 1 and 2, the geometry implication is to find the intersection of this line and $x = \delta$.



For algorithm 3, we plot the sorted thresholds for all bootstrap round with histgrams.

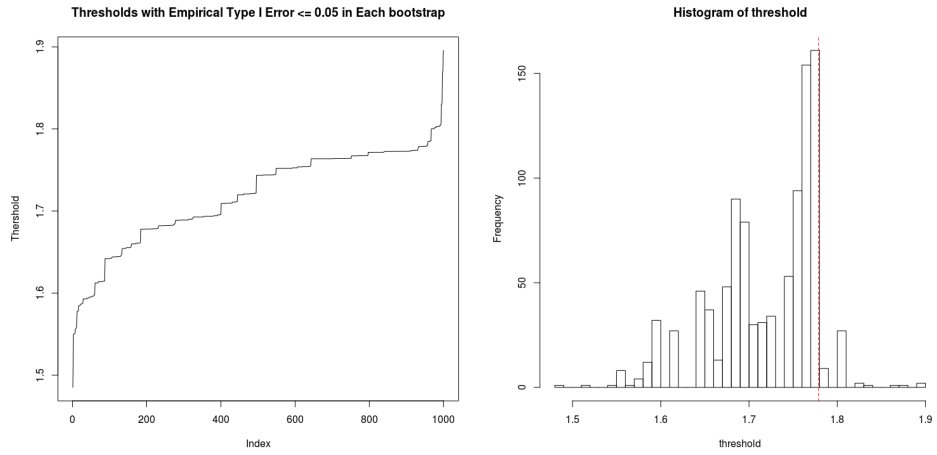


Figure 3: Sorted threshold for algorithm 3 and its CDF

4.3 Discussion

It is not surprising that bootstrap algorithm cannot bound the type-I error. However, it is still a good algorithm with plausible estimation for type-I error. We believe that by increasing the number of data and epoch of bootstrap, we will get a better result. However, it is still not a true bound, but a approximation.

5 Order NP Classification

5.1 Threshold Selection Procedure

We call this threshold selecting procedure as Order NP Classification just to distinguish from the first two procedures we discussed above. It is designed by Xin Tong, Yang Feng and Jingyi Jessica Li (refer to the acknowledgement). Our idea of implementing the bootstap in threshold selecting was inspired by this paper. The paradigm is to use only half of the class 0 data (size n) in the training set together with all the class 1 data to train the score function, then calculate the scores of the left-out class 0 observations and sort them to get a sequence of candidate thresholds $\theta_{(1)}, \ldots, \theta_{(n)}$, and the corresponding classifiers are $\phi_{(1)}, \ldots, \phi_{(n)}$. They made use of the properties of order statistics and found the probability that type I error $R_0(\phi_{(k)})$ is greater than α can be bounded by:

$$\mathbb{P}\Big[R_0(\phi_{(k)}) > \alpha\Big] \leq \sum_{j=k}^{n} \binom{n}{j} (1-\alpha)^j \alpha^{n-j}$$

Therefore, the upper bound depends only on α and k. The threshold selected by order NP classification is:

$$k^* = \min\{k \in \{1, \ldots, n\} : \sum_{j=k}^{n} \binom{n}{j} (1-\alpha)^j \alpha^{n-j}\}$$
$$\theta = \theta_{(k^*)}$$

5.2 NP-ROC

As stated in the previous parts, the emprical type I error can't represent the population type I error properly. Thus, choosing classification method according to traditional ROC can be misleading. Xin Tong etc. developed a counterpart for ROC under order NP classification called NP-ROC. In an NP-ROC plot, the horizontal axis is defined as the type I error upper bound with high probability $1 - \delta$. And the vertical band gives the upper and lower bound for (1- conditional type II error). Based on the properties of order statistic, NP-ROC gives a conservative estimation for type I and II error. The detailed construction procedure is included in their paper.

There are mainly two applications for this NP-ROC. If we don't have the type I error upper bound beforehand, we can select the x value with steepest rise in y on NP-ROC. In addition, we can draw the NP-ROC for different classification methods simultaneously on the same plot, and select the optimal methods with the highest power under the same type I error bound. In the application part, NP-ROC is used to choose the best classification methods in our experiment of data.

6 Simulation Comparison Result

6.1 Method

Comparison is based on the simulated data set: class 0 $(X|Y = 0) \sim N(0, 1)$, class 1 $(X|Y = 1) \sim N(2, 1)$, $\mathbb{P}(Y = 0) = \mathbb{P}(Y = 1) = 0.5$. $N = 1000$ data sets were generated, with $n = 1000$ observation in each set.

We set $\alpha = \delta = 0.05$.

In each data set for each method, we can obtain a classifier ϕ_i. Since it is one-dimensional, the classification scoring function is the identity function $f(x) = x$, and the classifier is $\phi(x) = I\{f(x) > \theta\}$.

We focus on finding the threshold θ, and evaluate their performance by calculating type I error violation rate:

$$V(\theta) = \frac{\#\text{of classifiers having population type I error} \geq \alpha}{\#\text{of classifiers}}$$

In this way, we run 500 times completely for each method. 500 thresholds with type-I errors are provided.

6.2 Result

We put these errors on the ROC curve to show the final violation rate. Three ROC curves are plotted in figure 5 and 6. The colored part are the scale of bootstrapped type-I error.

We perform our simulation on three methods. Left of figure 5 is the naive classical classifier. It simply consider the empirical error as population error, making the estimated type-I error to be a approximation of real population type-I error, which means that it does not bound the type-I error. The result shows the violation rate is about 0.5, which is reasonable.

On the right of figure 5 is the bootstrap classifier. By bootstrapping, violation rate decreases. However, it is still not bounded. The best result is NP classifier in figure 6. We can see that the violation rate is bounded perfectly.

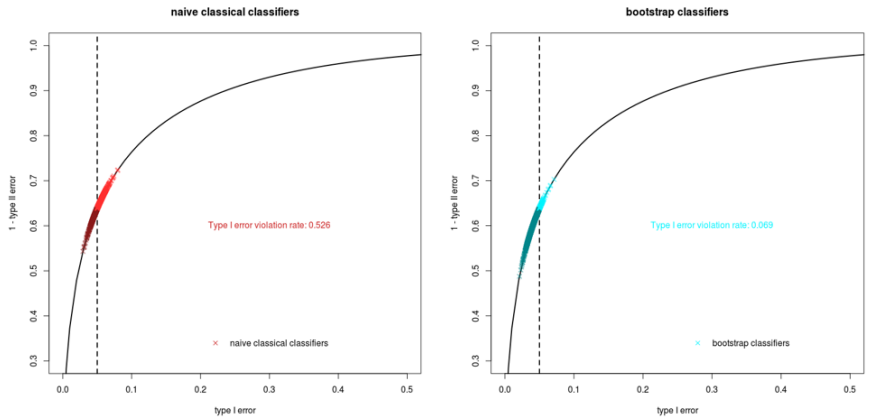


Figure 4: Sorted threshold for algorithm 3 and its CDF

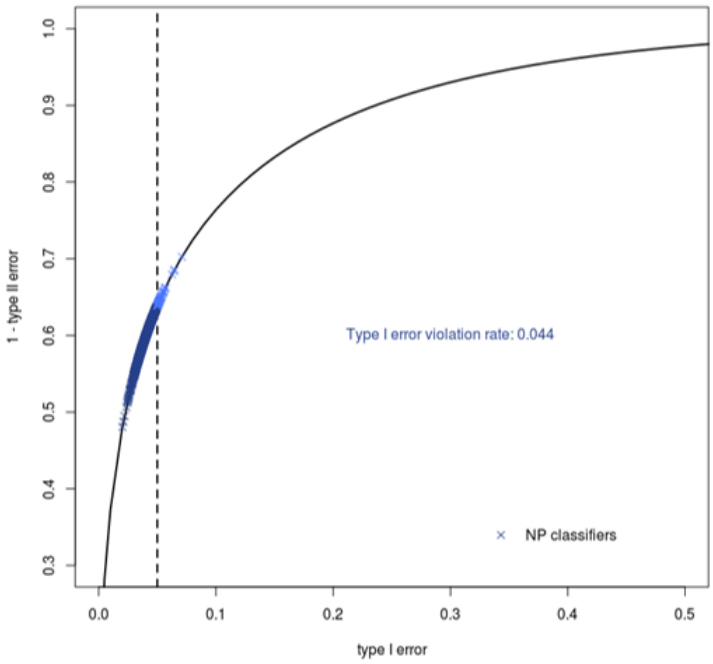


Figure 5: Sorted threshold for algorithm 3 and its CDF

7 Experimental Evaluation

In this part, we experiment with real data to demonstrate the advantage of NP classification method in application comparing with naive classification method.

In the experiment, we intend to solve a real-world business problem with a very unbalanced dataset as well as a specific need to control one type of error. After preprocessing the data and spliting 1/4 as testing set, we firstly select model and type I error upper bound based on NP-ROC bands, a comparison between NP method and naive empirical method for threshold selection is then presented.

7.1 Dataset introduction

In the introduction of a new product, a business organization want to promote it to its existing customers. For these customers, the company has a dataset with six features: card tenure, risk score, average balance, number of previous contacts, geographical group and customer type. In order to predict whether a customer will respond if contacted with regard to the new product, the company contacted a sample of 25,000 customers and collected their response status ($\{0, 1\}$ for {"non-respondent", "respondent"}). The objective is to build a binary classification model to select targets (class 1) for contacting from the pool of customers not contacted yet. The following is a summary on the raw dataset with complete observations:

Variable	Range	Mean	Median
Response	{0,1}	0.06857	0
Card Tenure	[0,641]	139.4	135
Risk Score	[520,791]	655.3	677
Previous contact	{0,1}	0.007	0
Average balance	[-132,10725]	3067	3240
Geographical group	{E,N,SE,W}	-	-
Residence type	{CN,CO,RE,SI,TO}	-	-

Table 5-1. Promotion Dataset with 22400 Obs.

In this dataset, only 6.9% observations are of class 0. This is a extremely unbalanced dataset, in which the more important class 1 is the rare class. Due to the dominating marginal probability of class 0, the classifier trained with the classical paradigm to minimize the overall risk would prioritize the type I error and possibly result in a large type II error (falsely classifying one as non-respondent). Whereas, it is generally not desirable in application, because the marginal cost of contact one more target is very small relative to the loss from missing a potential customer. In a word, we need to control type II type in this problem.

For the convenience of the discussion, we reverse the class labels for this data: $\{0, 1\}$ for {"respondent", "non-respondent"}, in order stick to the expression of controlling type I error. In the essence, type I error can be any type of error that misclassify the more important class, so this reversion does not cause any loss in generality.

As a result, in this experiment we attempt to bound type I error before minimizing over type II error, rather than minimize the overall error. The following sections address two tasks facing us: select a classification model to compute scores, and build a classifier with a threshold that controls type I error.

7.2 Model Selection with NP-ROC Plot

With a specific need to control type I error, there are two immediate challenges in building a classification model: 1) choose a classification algorithm with optimal prediction power given our restriction on type I error; 2) set a restriction on type I error such that it is sufficiently small, while at the same time not too small to maintain a considerable prediction power. NP-ROC plot is a powerful visual aid to make those two decisions simultaneously.

As introduced in Section 4-4, NP-ROC plot displays the prediction power (1-type II error) of different classification algorithms over different choices of type I error upper bound in NP classification paradigm. With a violation tolerance rate $\delta = 0.1$, Figure 5-1-1 compares 7 common classification algorithms in the NP-ROC plote, while an AUC plot (Figure 5-1-2) derived from these NP-ROC bands

shows that adaboost has both the highest lower bound and the highest upper bound for classification power in general.

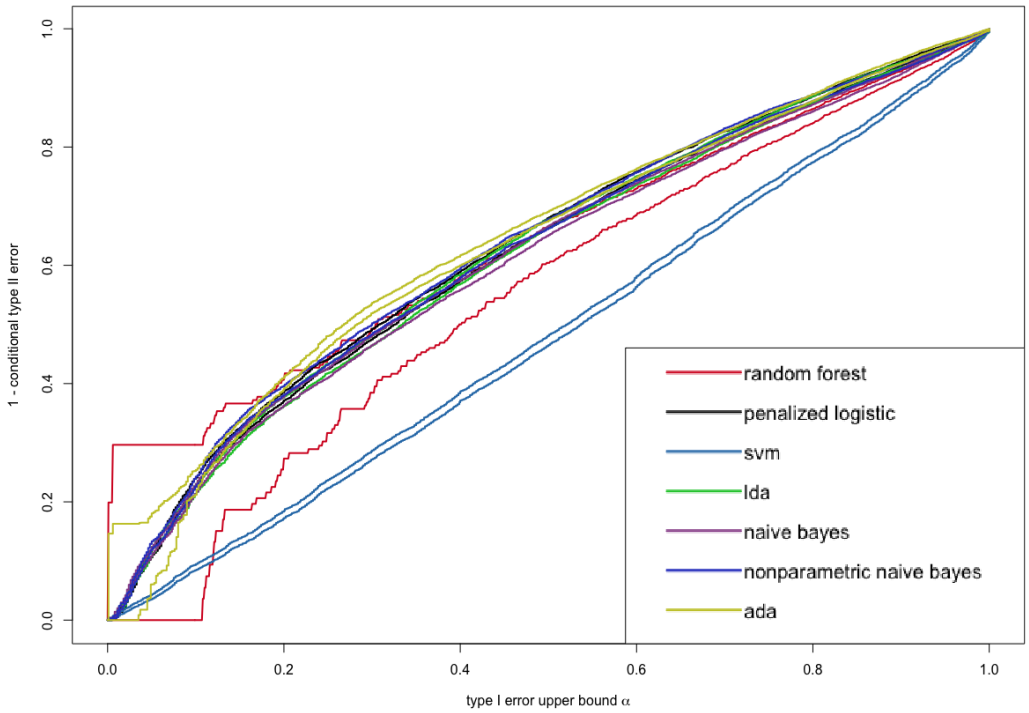


Figure 6: NP-ROC

Figure 5-1-1. NP-ROC plot for 7 classification methods

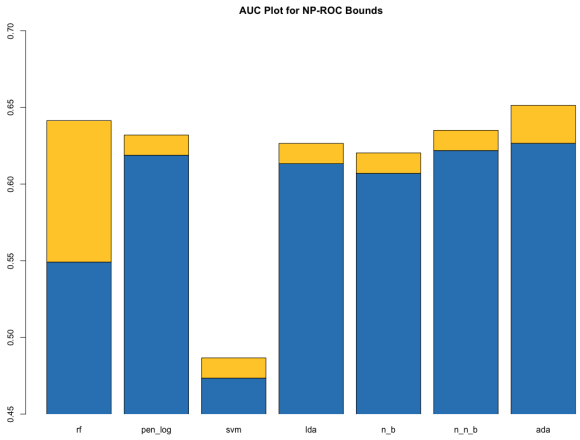


Figure 5-1-2. AUC plot for 7 classification methods

As we are interested in a small type I error bound, we further compare two methods that seem to yield relatively higher power - adaboost and random forest - in Figure 5-2. From this plot it can be observed that a too small type I error upper bound could lead to great loss in power, i.e., the learned classifiers

make all class 1 predictions to avoid type I error. Thus, a reasonable choice for this problem should be adaboost algorithm with a 0.2 type I error upper bound.

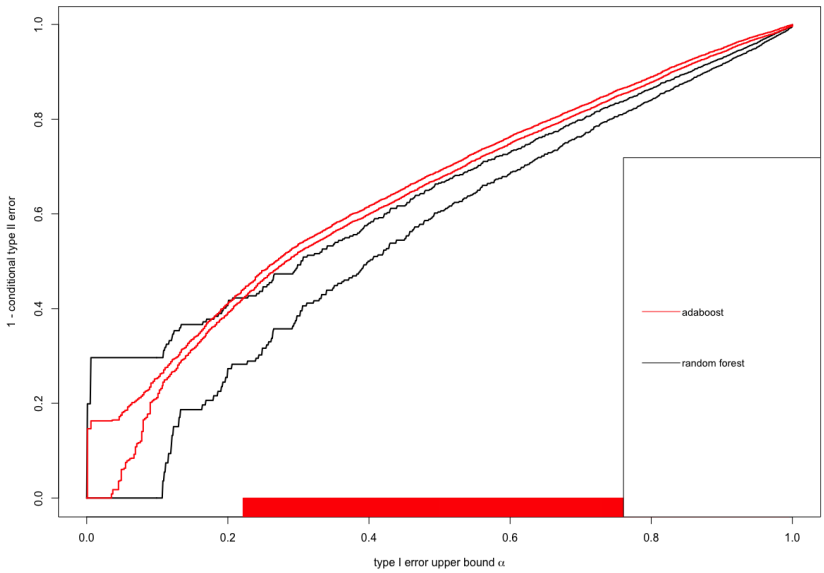


Figure 7: NP-ROC

Figure 5-2. NP-ROC for Adaboost and Random forest

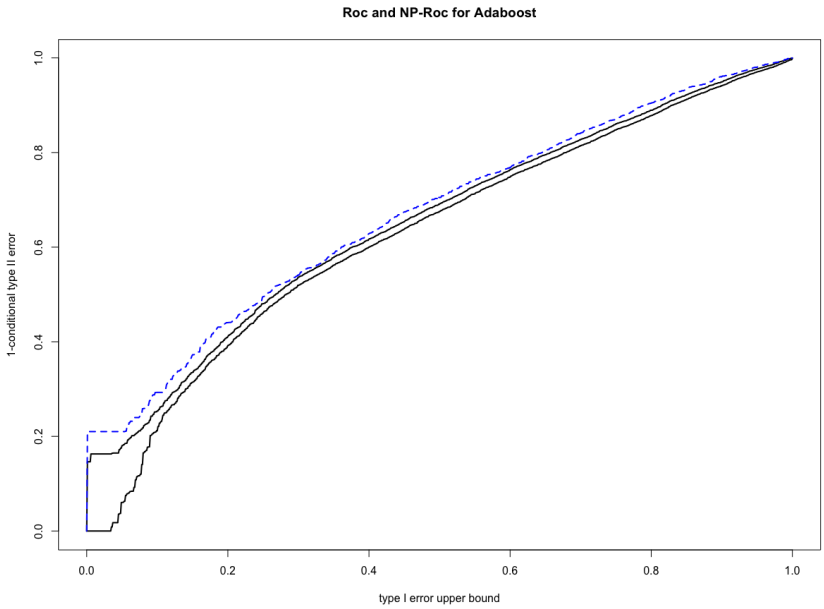


Figure 5-3. NP-ROC and empirical ROC for Adaboost

7.3 Threshold Selection with NP Method and Naive Method

Using adaboost as our classification model, we may compute the scores for all training data. In order to control over type I error, we need to select a threshold based on rank statistics corresponding to the scores of all class 0 observations. By applying the NP algorithm introduced in section 4, we choose

$\theta_1 = 0.0432$ as the classification threshold. For comparison, we also choose the 80% quantile of class 0 scores $\theta_2 = 0.099$ with naive method.

The comparison of these two models on the test set is presented in Table 5-2. NP method gives a significantly better prediction, with a type I error of bounded by the upper bound 0.2, and also a higher overall accuracy. As discussed before, θ_1 based on NP paradigm bound the population error at a $1 - \delta = 0.9$ confidence level, whereas the naive method based on empirical error fail to bound the error.

Method	NP method ($\delta = 0.1$)	Naive method
Type I error	0.1313131	0.4343434
Type II error	0.6112821	0.7971795
Accuracy	0.4226829	0.1999047

Table 5-2. Testing errors of two methods ($\alpha = 0.2$)

It is also notable that NP method also achieves a higher accuracy than naive method. It can be accounted for by the advantage of NP classification paradigm over biased data. Due to the dominating marginal probability of class 1, the model tend to boost fit in training by making more class 1 predictions, resulting in an empirical type II error lower than population. As demonstrated in Figure 5-3, the empirical ROC line in dashed line exaggerates the prediction power and is higher than the population upper bound.

8 Conclusion

In this project, we start from a basic problem in binary classification application — controlling tpye I error. We propose a straightforward method through bootstrap and replicate NP classifier method in the paper.

The result shows that bootstrap method cannot bound the type-I error. However, it is still a good algorithm with plausible estimation for type-I error. We believe that by increasing the number of data and epoch of bootstrap, a result closer to the truth can be obtained. However, it is still a method inferior to NP classifcation paradigm, as it is an approximation intseand of true bound.

On the other hand, by implementing the NP classification method, its advantages is proven in solving real-world problems. Firstly, the construction of NP-ROC plot with population bounds for prediction power provides a intuitional tool to select classification model and type I error upper bound simultaneously. Secondly, NP algorithm bound the population type I error instead of empirical error, which yields a more precise control over type I error. Last but not least, NP method is especially useful for very unbalanced datasets, where it controls over population error to reduce loss of accuracy from bias in training sample.

8.1 Future Plan

Bounding the type-I error is a vital concern in real-life classification problems. The NP classifier is a plausible method with proven strength, especially with the possibility to be generalized to multi-class task. However, the type II error would usually be infated greatly with NP classification method. Thus, the result can be too conservative in some cases.

For the bootstrap part, it is still unknown whether there exist a way to bound the type-I error. There are still lots of works to do.

Acknowledgments

Xin Tong, Yang Feng, and Jingyi Jessica Li. "Neyman-Pearson (NP) Classification Algorithms and NP Receiver Operating Characteristics (NP-ROC)."

Kaggle Data Set|Promotion response and target datasets [https://www.kaggle.com/regivm/promotion-response-and-target-datasets]