UNIVERSITY OF CALIFORNIA
Los Angeles

SMOTE Variants for Imbalanced Binary Classification: Heart Disease Prediction

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Statistics

by

Xiaoru Zheng

2020

© Copyright by
Xiaoru Zheng
2020

ABSTRACT OF THE THESIS

SMOTE Variants for Imbalanced Binary Classification:
Heart Disease Prediction

by

Xiaoru Zheng
Master of Science in Statistics
University of California, Los Angeles, 2020
Professor Jingyi Li, Chair

Class imbalance is prevalent in many medical diagnosis problems, where the number of patients suffering from a particular disease is much smaller than the number of healthy people in the population. A similar phenomenon also occurs in credit card fraud detection and spam email filtering. Approaches to dealing with imbalanced data can be divided into those at data and algorithm levels. At the data level, resampling techniques such as oversampling and undersampling can result in a more balanced distribution of classes. At the algorithm level, cost-sensitive learning takes the prediction error into account in the training process, which can achieve better overall prediction performance. In this study, variants of the Synthetic Minority Oversampling Technique (SMOTE) are implemented for comparison: regular SMOTE, Borderline-SMOTE, SVM-SMOTE, KMeans-SMOTE, which are combined four classification algorithms under the classical and Neyman-Pearson (NP) paradigms to build predictive models on the heart disease data, and the performance of these models is compared. Our results show that the SVM-SMOTE and the Borderline-SMOTE outperform other SMOTE variants, and the NP classification is superior in controlling the type I error effectively.

The thesis of Xiaoru Zheng is approved.

Hongquan Xu

Chad J Hazlett

Jingyi Li, Committee Chair

University of California, Los Angeles

2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

For classification problems, standard machine learning algorithms assume that the proportion of different classes in the actual dataset is roughly equal. However, in many practical applications, classes of the response variable are not presented in an equal proportion. The objective of a clinical test is usually to identify minority observations that are underrepresented in the dataset. Predictive models built with imbalanced data are often biased towards predicting observations as belonging to the majority class. It happens because the model might appear to have high accuracy by favoring the majority class. The importance of accurately classifying the minority class is especially critical when there is a high cost for misclassifying minority instances, such as misdiagnosing a rare but significant disease. In other words, incorrectly classifying a diseased patient as having no disease is more costly than the reverse.

There are two commonly used methods to tackle the problem of under-representation of minority cases. The first approach is to resample the imbalanced data such as oversampling, undersampling, or a hybrid of the two. The second approach is to increase the weight of mispredicting a minority instance in the cost function[1][2], such as logistic regression, random forest, adaboost, and support vector machine (SVM). To be noted, the Neyman-Pearson (NP) classification paradigm, which belongs to the second approach, can tackle the optimization problem on controlling the population type I error when class 0 refers to the minority class[3]. When class 0 refers to diseased patients, the type I error has a higher cost than the type II error, since the type I error may result in a tragic loss of life when a patient is tested negative for a disease, while in reality, the patient does have that disease. The NP classification develops an umbrella algorithm that builds on scoring-type base classification algorithms

such as logistic regression and random forest.

Accuracy is a commonly used metric to evaluate the performance of a model. However, it has been largely emphasized that accuracy may mislead results on the imbalanced data[4], and more appropriate metrics are derived from the confusion matrix, such as precision and recall. Intuitively, precision is a measure of correctness (i.e., out of positively-labeled instances, how many are truly positive), and recall (or sensitivity) is a measure of completeness or accuracy of positive instances (i.e., how many instances of the positive class are labeled correctly/positively). Precision and recall share an inverse relationship. Another measurement to assess the model performance is specificity, or true negative (TN) rate (i.e., how many instances of the negative class are labeled correctly/negatively). The Receiver Operating Characteristic (ROC) curve summarizes the model performance over a range of trade-offs or thresholds between sensitivity and 1-specificity[5]. The Precision-Recall (PR) curve shows precision values for corresponding sensitivity values, and its baseline moves with the class distribution. In other words, the PR curve is sensitive to the ratio between positive and negative instances. Like the area under the ROC curve (AUROC), the area under the PR curve (AUPRC) often serves as an evaluation criterion of a model, and it is approximated by the average precision (AP)[6].

This study addresses the oversampling method, specifically the Synthetic Minority Oversampling Technique (SMOTE), and how SMOTE variants differ. Chawla et al. (2002) proposed the SMOTE as a resampling method to oversample the minority class, and the main idea was to generate synthetic minority instances surrounding existing minority instances[7]. There are multiple variations of SMOTE: Borderline-SMOTE, SVM-SMOTE, KMeans-SMOTE, SMOTE-Nominal Continuous (SMOTE-NC). These methods aim to overcome the weakness of the original SMOTE algorithm. For instance, Borderline-SMOTE addresses to only oversample borderline instances of the minority class because these instances are more likely to be misclassified. SMOTE-NC method is used when the data contains both numerical and categorical features. We will explain the Borderline-SMOTE, SVM-SMOTE, and Kmeans-

SMOTE in Chapter 2 because the dataset we used only contains numerical features. After oversampling the minority class by SMOTE variants, predictive models are built with multiple classification algorithms: logistic regression, random forest, adaboost, and SVM, all of which are implemented under the classical and the NP paradigms. We will use multiple criteria, including accuracy, sensitivity, specificity, F1-score, ROC curve and PR curve to evaluate the model performance.

CHAPTER 2

Methods

2.1 Synthetic Minority Oversampling Technique (SMOTE)

Synthetic Minority Oversampling Technique (SMOTE) is a well-known oversampling technique for imbalanced data classification problems at the data preprocessing step[7]. This method is motivated by a successful technique in handwritten character recognition that creates the extra training data by performing specific operations, such as rotating handwritten characters[8]. SMOTE generates synthetic instances using the linear interpolation within the minority class, and it effectively forces the region of the minority class to become broader. Briefly, SMOTE firstly selects a minority instance randomly and finds the k-nearest neighbors of that instance. A synthetic instance is then created at a randomly selected point on the line connecting the instance and one random instance among the k neighbors in the feature space.

For a minority instance, the number of randomly chosen neighbors among its k-nearest neighbors depends on the amount of oversampling instances required[7]. For example, with $k = 5$, if the needed amount is three times the size of the original data, three neighbors out of five nearest neighbors will be chosen, and one synthetic instance is generated on each line connecting the minority instance and the selected neighbors. The following steps can be used to create a synthetic instance:

1. Pick an instance from the minority class randomly.
2. Find the k-nearest neighbors of the current instance.

3. Take the difference between the current instance and one of the neighbors, then multiply the difference by a random number *gap* between 0 and 1.

4. Add the result from the previous step to the current instance and create a new synthetic instance.

We can write each j-th feature, $j = 1, \ldots, p$, of the synthetic instance $X_{\text{synth}} = [x^{(1)}_{\text{synth}}, x^{(2)}_{\text{synth}}, ...x^{(p)}_{\text{synth}}]$, which can be written as[9],

$$x^{(j)}_{\text{synth}} = x^{(j)} + gap^{(j)} \times [x^{(j)}_{\text{neighbor}} - x^{(j)}]. \tag{2.1}$$

where:

- $x^{(j)}$ is the j-th feature of the chosen instance x
- $x^{(j)}_{\text{neighbor}}$ is the j-th feature of a randomly chosen neighbor of the instance x
- $gap^{(j)}$ is the uniformly distributed random variable from (0, 1) for the j-th feature

Figure 2.1 visualizes how SMOTE works, where only a part of the simulated instances is plotted. This figure shows that x_i is one of the selected instances from the minority class and connected to its five nearest neighbors by lines. After a neighbor x_{neighbor} is randomly selected, the synthetic instance x_{synth} is generated along the line joining x_i and x_{neighbor}.

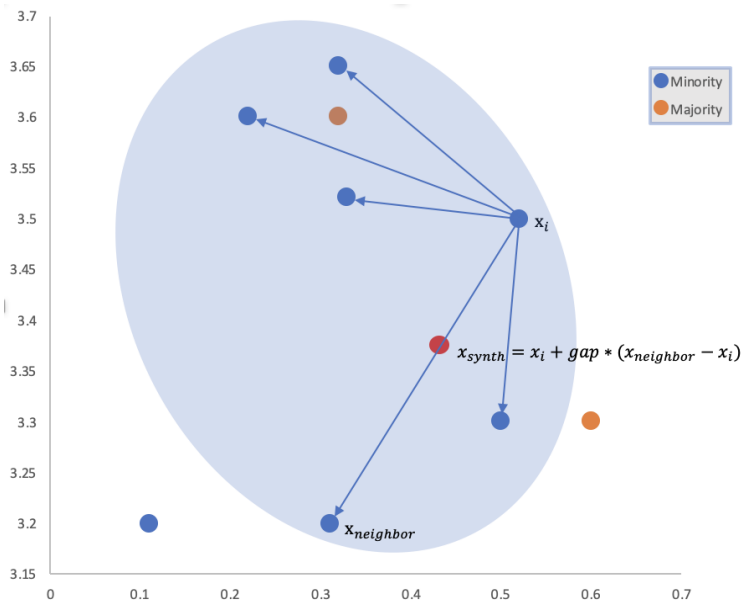

Figure 2.1: Generating a synthetic instance for the minority class with SMOTE

Two theoretical properties of SMOTE are summarized below: (1) a SMOTE instance has the same expected value as that of a minority instance; (2) a SMOTE instance has a smaller variance than that of a minority instance[10]:

$$E(X^{SMOTE}) = E(X), \quad Var(X^{SMOTE}) = \frac{2}{3}Var(X). \tag{2.2}$$

2.1.1 Borderline-SMOTE

Han et al. (2005) proposed the Borderline-SMOTE1 and Borderline-SMOTE2 methods by oversampling the minority class borderline instances only[11]. In the training process, Borderline-SMOTE algorithms attempt to learn the borderline of each class, where these borderline instances and the ones nearby are more likely to be misclassified than the ones far from the borderline. In Borderline-SMOTE, all instances in the minority class are divided into three groups: NOISE, DANGER, and SAFE.[12].

- NOISE instances are rare, probably incorrect, located in areas occupied by the majority class instances.

- DANGER instances are placed in the area near class boundaries and typically overlap with majority instances.
- SAFE instances are relatively easier to recognize, and they are the main representatives of the minority class.

Figure 2.2 can help us better to understand the above three groups in the minority class.

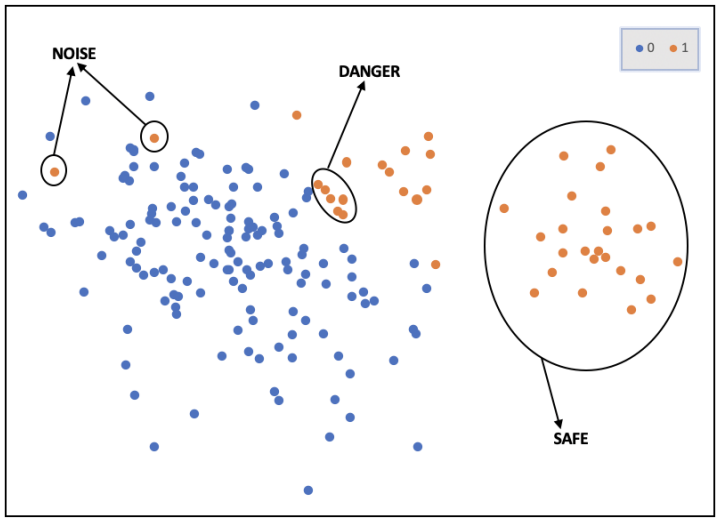

Figure 2.2: Example of imbalanced data

After all instances of the minority class are categorized, synthetic instances are then created along the line between DANGER instances and their nearest neighbors. Below is the detailed Borderline-SMOTE1 algorithm and the definitions of the three groups. Supposing the training set is T, the minority class is P, and the majority class is N, we write the two classes of instances as

$$P = \{p_1, p_2, ..., p_{pnum}\},\ N = \{n_1, n_2, ..., n_{nnum}\}$$

where $pnum$ and $nnum$ are the numbers of minority and majority instances, respectively. The following steps of Borderline-SMOTE1 is adapted from Han et al. (2005)[11].

Step 1. For every p_i (i=1,2,...,$pnum$) in the minority class P, calculate its m nearest neighbors from the whole training set T. The number of majority instances among m nearest neighbors is denoted by m', where $0 \leq m' \leq m$.

Step 2. Categorize all minority instances into three groups by the following criteria:

- If $m' = m$, i.e., all the m nearest neighbors of p_i are majority instances. p_i is considered to be in the set NOISE, and it will not be operated in the following steps.
- If $\frac{m}{2} \leq m' < m$, the number of p_i's majority nearest neighbors is larger than the number of its minority ones, p_i will be misclassified easily. Then, this p_i should be included in the set DANGER.
- If $0 \leq m' < \frac{m}{2}$, p_i will be considered in the set SAFE and not operated in the following steps.

Step 3. Since all instances in set DANGER are the borderline points of the minority class P, we can see that DANGER $\subseteq P$. We set

$$\text{DANGER} = \{p'_1, p'_2, ..., p'_{dnum}\}, \quad 0 \leq dnum \leq pnum.$$

For each instance in set DANGER, calculate its k nearest neighbors from P.

Step 4. Generate $s \times dnum$ synthetic minority instances from the instances in set DANGER, where s is an integer between 1 and k. For each p'_i, randomly select s nearest neighbors from its k nearest neighbors in P. Then perform the following procedure which is similar to the regular SMOTE algorithm:

4.1. Calculate the difference dif_j $(j = 1, 2, ..., s)$ between p'_i and its s nearest neighbors from P. Then multiply the difference by a random number r_j $(j = 1, 2, ..., s)$ between 0 and 1.

4.2. Add the result from 4.1 to p'_i, and create a synthetic instance:

$$synthetic_j = p'_i + r_j \times dif_j, \quad j = 1, 2, ..., s. \tag{2.3}$$

Repeat the above procedure for each p'_i in set DANGER to attain $s \times dnum$ synthetic instances.

Han et al. (2005) also introduced a variation of the Borderline-SMOTE1 called Borderline-SMOTE2[11]. Borderline-SMOTE2 generates synthetic instances not only from the set DANGER and its nearest neighbors in P, but also from its nearest neighbors in N. In Borderline-SMOTE1, the difference dif_j $(j = 1, 2, ..., s)$ between p'_i and its s nearest neighbors in P is multiplied by a random number r_j between 0 and 1, while in Borderline-SMOTE2, this random number r_j ranges from 0 to 0.5. Therefore, the newly created synthetic instances are closer to the minority class.

2.1.2 SVM-SMOTE

Nguyen et al. (2011) proposed an alternative method of the Borderline-SMOTE by using the support vector machine (SVM) algorithm instead of the k-nearest neighbor algorithm to identify the misclassified instances on the decision boundary. This method is called SVM-SMOTE[13]. The SVM is used to locate the decision boundary defined by support vectors, and a new instance will be randomly created along the lines joining each minority class support vector with the number of its nearest neighbors using extrapolation (Fig. 2.3a) or interpolation (Fig. 2.3b), depending on the density of the majority class instances around it.

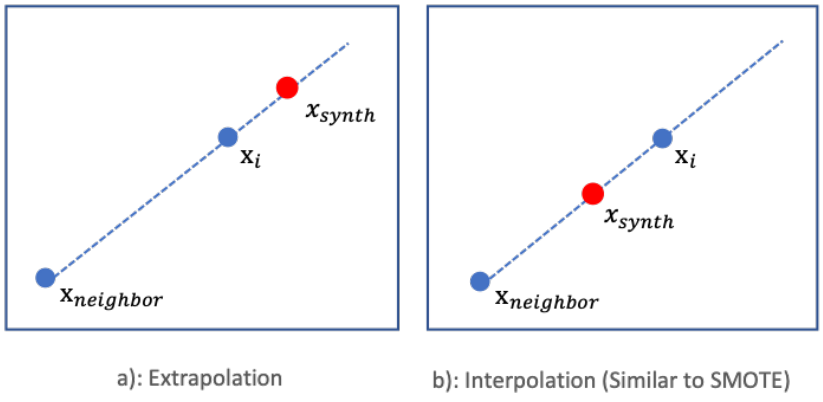

Figure 2.3: Utilize extrapolation (a) and interpolation (b) to generate synthetic instances

Denote $\mathbf{SV}^+$, where $\mathbf{sv}_i^+ \subseteq \mathbf{SV}^+$, as the set of the minority class support vectors, $\mathbf{nn}$ as the array contains k minority nearest neighbors of each $\mathbf{sv}_i^+$, and *amount* as the array contains the amount of synthetic minority instances corresponding to each $\mathbf{sv}_i^+$. For each $\mathbf{sv}_i^+ \subseteq \mathbf{SV}^+$, calculate its m nearest neighbors from the training set. The number of majority instances among m nearest neighbors is denoted by m'. The two conditions of generating new instances are the following:

- If $0 \leq m' < \frac{m}{2}$, meaning that fewer than a half of the m nearest neighbors of p_i' are majority instances, along the lines joining $\mathbf{sv}_i^+$ with its k minority nearest neighbors (in the first to k-th nearest neighbor order), create $amount[i]$ instances using extrapolation to expand the minority class area by the following equation:

$$synthetic_i = \mathbf{sv}_i^+ + \rho \times (\mathbf{sv}_i^+ - \mathbf{nn}[i][j]), \tag{2.4}$$

  where $\mathbf{nn}[i][j]$ is the j-th positive nearest neighbor of $\mathbf{sv}_i^+$, and ρ is a random number in the range $[0, 1]$.

- If $\frac{m}{2} \leq m' < m$, this is the case when the nearest neighbors of selected $\mathbf{sv}_i^+$ are mostly majority class. Then the current boundary area of the minority class will be consolidated or interpolated in a similar way to SMOTE, while the new instances will be generated in the order of the first to k-th nearest neighbor instead of randomizing the selection of nearest neighbors:

$$synthetic_i = \mathbf{sv}_i^+ + \rho \times (\mathbf{nn}[i][j] - \mathbf{sv}_i^+). \tag{2.5}$$

The main characteristic of SVM-SMOTE is the use of extrapolation to expand the minority class region, when there are fewer majority class instances. Thus, this algorithm helps to increase the chance to see minority class instances near the ideal boundary.

2.1.3 Kmeans-SMOTE

Kmeans-SMOTE, which employs the Kmeans clustering algorithm in conjunction with the SMOTE to rebalance the dataset, was proposed by Last et al., (2017)[14]. Kmeans-SMOTE

does only avoid the generation of the noise by oversampling in the safe area but also pays attention to both between-class and within-class imbalance. The Kmeans-SMOTE algorithm consists of three steps: clustering, filtering, and oversampling:

Step 1: *Clustering*. Cluster the instances into k groups by using the Kmeans clustering.

Step 2: *Filtering*. Select clusters to be oversampled and determine the number of new instances to be generated in each cluster. The cluster selection is based on the proportion of minority and majority instances in each cluster. By default, any clusters with at least 50% of minority instances are selected for oversampling. Then sampling weights are assigned to the selected clusters to determine the instances to be generated. The detailed computation of the sampling weights of clusters was reported in a previous study[14].

Step 3: *Oversampling*. Use SMOTE to oversample each selected cluster. Choose a random minority instance x_i within a cluster, find one of its random minority neighbor x_{neighbor}, and determine a new instance x_{synth} by interpolating x_i and x_{neighbor}. The process will be repeated until reaching the number of minority instances, which is prespecified.

Kmeans-SMOTE clusters the entire data set regardless of the class label. Specifically, Kmeans clustering enables the discovery of overlapping class regions and avoids oversampling in safe areas. Kmeans-SMOTE is distinct from other SMOTE variants because it distributes instances according to the cluster density, which is based on the cluster size, and this is an effective way to combat the within-class imbalance.

2.2 Machine Learning Methods for Binary Classification

2.2.1 Classical Classification Algorithms

Logistic regression

Logistic regression is a generalization of linear regression for binary classification, which is used to model the probability of a particular event or class with a logistic function[15].

Suppose $X \in \mathbb{R}^{n \times p}$ is an n by p matrix which contains n observations and p features, x_i^T is a p-dimensional row vector containing p features of the i^{th} observation, and β is a p-dimensional column vector containing features' coefficients. $Y \in \{0, 1\}$ represents the binary outcome and $Y_i \sim Bernoulli(\pi_i)$. The linear relationship between the features and the log-odds of the event when $Y_i = 1$ can be written in the following form:

$$l = \log\left[\frac{P(Y_i = 1)}{1 - P(Y_i = 1)}\right] = \log(\frac{\pi_i}{1 - \pi_i}) = x_i^T \beta. \tag{2.6}$$

The likelihood function of $\pi = (\pi_1, \ldots, \pi_n)^T$ is used to estimate model coefficients β. It is the joint probability mass function of n Bernoulli variables.

$$L(\pi) = f(y_1, ..., y_n) \overset{ind}{=} \prod_{i=1}^{n} f(y_i|\pi_i) = \prod_{i=1}^{n} \pi_i^{y_i}(1 - \pi_i)^{(1-y_i)}. \tag{2.7}$$

Logistic regression with regularization adds a constraint (or regularizer) to the loss function to control the excessively fluctuating function. Thus, coefficients do not take extreme values, making the model less complicated and less prone to overfitting. For the regularization term, Ridge[16], Lasso (Least Absolute Shrinkage and Selection Operator)[17], and Elastic net[18] are commonly used.

Random forest

Random forest is a tree-based ensemble machine learning method using the bagging, or bootstrap aggregation technique to build an extensive collection of decorrelated decision trees[19]. In random forest, each tree splits out a class prediction, and the class with most votes will become the final prediction. Random forest can be improved by bagging because it can decorrelate the trees by introducing the split of a random subset of all features in the data. Thus, random forest generally exhibits a substantial performance improvement over the single tree classifier.

Random forest algorithm is as follows:

Step 1. Set the total number of trees as B.

For $b = 1$ to B:

(a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

(b) Grow a random forest tree T_b on the bootstrapped sample by recursively repeating the following steps for each terminal node of a tree, till the minimum node size n_{min} is reached.

i. Select m features at random from p features (Typically $m = \sqrt{p}$).

ii. Pick the best features/split-point among m.

iii. Split the node into two daughter nodes.

2. Output the ensemble of trees T_b. For a classification problem, the final prediction will be the majority voting results in B trees.

Adaboost

Adaboost, short for "Adaptive Boosting", is another ensemble method that uses the boosting framework to combine multiple weak classifiers into a stronger classifier[20]. A weak classifier $h(x)$ with $\Pr[h(x) = y] = 0.5 + \epsilon$, performs slightly better than a random guessing, where y is the true label and ϵ is a very small constant. Boosting is a sequential process that retrains the algorithm iteratively by choosing the training set based on the accuracy of the previous training process.

Given a data set containing n instances and p features, $x_i \in \mathbb{R}^p$, $y_i \in \{-1, 1\}$, $i = 1, ..., n$.

Initialize the uniform weight for each instance as:

$$D_0(i) = \frac{1}{n}, i = 1, ..., n. \tag{2.8}$$

For iteration $t = 1, ..., T$:

- Train a weak classifier $h_t(x) \in \{-1, 1\}$ that minimize the weighted error $\epsilon_t = \frac{1}{n} \sum_{\substack{i=1 \\ h_t(x_i) \neq y_i}}^{n} D_t(i)$.

- Choose $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$
- Update the weight for each instance:

$$D_{t+1}(i) = \frac{D_t(i)\exp\left(-y_i\alpha_t h_t(x_i)\right)}{Z_t}, \tag{2.9}$$

where Z_t is a normalization factor that ensures $\sum_{i=1}^{n} D_{t+1}(i) = 1$

After learning, the final classifier is based on a linear combination of weak classifiers:

$$H(x) = \text{sign}\Big[\sum_{t=1}^{T}\alpha_t h_t(x)\Big]. \tag{2.10}$$

Adaboost is a greedy algorithm that builds up a "strong classifier" $H(x)$ incrementally by optimizing the weight of weak classifiers at every time.

Support vector machine

The objective of the support vector machine (SVM) algorithm is to find a hyperplane in a high dimensional space that separates two classes[21]. A hyperplane is a decision boundary with the maximum margin, i.e., the distance from the hyperplane to the nearest instance. There are multiples types of hyperplane learning methods based on different types of the used kernels. For example, the dot product is the distance measure used in linear kernel SVM. Some more complex kernels, such as the polynomial kernel and radial kernel, allow lines to separate the classes that are curved or even more complex.

Given a training set n of $\{x_i, y_i\}_{i=1}^{n}$, where $x_i \in \mathbb{R}^p$, p is the number of features, $y_i \in \{-1, +1\}$ is the i^{th} output, the SVM classifier is defined as:

$$f(x) = \text{sign}\Big[\sum_{i=1}^{n}\alpha_i y_i K(x, x_i) + b\Big]. \tag{2.11}$$

where α_i are positive real constants, and b is a real constant. $K(x, x_i)$ is the choice of a kernel.

2.2.2 Neyman-Pearson (NP) Classification Umbrella Algorithm

Neyman-Pearson (NP) classification umbrella algorithm was proposed by Tong et al., (2018)[3], which implements the NP paradigm for all scoring-type classification algorithms such as logistic regression, random forest, and support vector machine. The umbrella algorithm aims to find the smallest threshold for base classifiers so that the violation rate is controlled under a prespecified tolerance rate δ. The violation rate is the probability that the type I error exceeds α, where α is the type I error upper bound.

Tong et al. denote a classifier $\varphi(\cdot)$ as a mapping $\varphi : \mathcal{X} \to \{0, 1\}$ that returns the class prediction of a given x, and $x \in \mathbb{R}^p$ is an instance with p features. They also denote the classification scores of a trained base classification algorithm applied to a left-out class 0 instance set (with n instances) as $T_1, ..., T_n$, and the kth order statistic $T_{(k)}$ as $T_{(1)} \leq ... \leq T_{(n)}$. The theoretical foundation of the umbrella algorithm is explained as follows.

Step 1: Given n, α, and δ, calculate the rank threshold $k^* = \min\{k \in \{1, ..., n\} : v(k) \leq \delta\}$, where $v(k) = \sum_{j=k}^{n} \binom{n}{j}(1-\alpha)^j \alpha^{n-j}$ is the upper bound probability that the type I error of $\hat{\varphi}_k$, which denotes a trained base classifier with $T_{(k)}$ as the threshold, exceeds α.

Step 2: Divide the training set into two halves, each of size n. The first part contains the data from both classes (0 and 1) that is used to train a base classification algorithm (denoted by $f(\cdot)$), and the second part is used as the left-out class 0 instances, $x_1, \ldots, x_n$. Apply the base classification algorithm $f(\cdot)$ to the second half data to get classification scores $T_1, ..., T_n$, i.e. $f(x_i) = T_i$ for $i = 1, ..., n$. Then rank them by increasing order $T_{(1)} \leq ... \leq T_{(n)}$.

Step 3: Construct a classifier $\hat{\varphi}_{k^*}(x) = \mathbb{1}(f(x) > T_{(k^*)})$, where k^* is the rank threshold from step 1.

The above steps demonstrates the procedure for one random split of the training data ($M = 1$). For multiple ($M > 1$) random splits, with each split dividing the training data into two halves, we can construct an ensemble NP classifier by majority voting.

2.3 Model Evaluation Metrics for Binary Classification

2.3.1 Confusion Matrix

The model performance for a classification problem is typically evaluated by a confusion matrix, where the number of correct and incorrect predictions are summarized in a count value by each actual class, as illustrated in Table 2.1. In this study, 'positive' represents the majority class or class 1, and 'negative' denotes the minority or class 0. TP and TN indicate the number of positive and negative instances that are classified correctly, and FN and FP denote the number of misclassified positive and negative instances, respectively. The model accuracy can be calculated from the confusion matrix.

Table 2.1: Confusion matrix for binary classification

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+FN+FP+TN}} \tag{2.12}$$

However, when evaluating the model performance for imbalanced data, accuracy generally leads to a classifier that predicts the majority class better, while behaving poorly on the minority class. Even when all majority instances are correctly predicted, and all minority cases are misclassified in the highly imbalanced data, the model accuracy can still be very high. Thus, accuracy cannot can not reflect reliable predictions for the minority class, and more reasonable evaluation metrics are needed.

Sensitivity, specificity, precision, and F-measure

Sensitivity, or recall, the same as the TP rate, is the ratio of correctly predicted positive observations to the observations in the actual positive class. In our example, the question that sensitivity answers is: of all patients who are truly positive, or "undiseased", how many of them are actually predicted/labeled positive. Specificity, or TN rate, is the proportion of diseased patients that are correctly identified as diseased.

$$\text{Sensitivity} = \text{Recall} = \text{TP rate} = \frac{\text{TP}}{\text{TP+FN}} \tag{2.13}$$

$$\text{Specificity} = \text{TN rate} = \frac{\text{TN}}{\text{TN+FP}} \tag{2.14}$$

Precision measures the fraction of instances classified as positive that are truly positive. The question that precision answers is: of all patients that are labeled as positive, or tested "undiseased", how many of them are truly undiseased.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{2.15}$$

F-measure is the weighted harmonic mean of precision and recall, and it takes both false positive and false negative cases into account when evaluating the model performance. F-measure is high when both recall and precision are high. It can be adjusted by a coefficient of β, where β represents the relative importance between precision and recall. For instance, F_1 ($\beta = 1$) puts the balanced importance on the precision and recall.

$$F_\beta = (1 + \beta^2)\frac{\text{Recall} \cdot \text{Precision}}{\beta^2 \cdot \text{Precision} + \text{Recall}} \tag{2.16}$$

2.3.2 ROC Curve, NP-ROC Bands, and Precision-Recall Curve

ROC curve

The relationship between sensitivity and 1-specificity (FP rate) can be visualized on a Receiver Operating Characteristic (ROC) curve space where the TP rate (sensitivity) is plotted on

the y-axis and FP rate (1-specificity) on the x-axis.

ROC curve is based on the notion of a "separator" (or decision) value on which results for the 'disease' and 'non-disease' form a pair of overlapping distributions[5]. Figure 2.4 and Figure 2.5 illustrate the concept of the ROC curve. Figure 2.4 shows two overlapping distributions with four thresholds for 'undiseased' (red curve) and 'diseased' (green curve). Figure 2.5 shows the corresponding ROC curve plot. In the probability density plot, as the threshold moving from left to right side, the TP rate (sensitivity) will increase, while the TN rate (specificity) will decrease. This can also be seen from the ROC curve when the value of the threshold increases, that the sensitivity will increase, while the specificity will decrease.

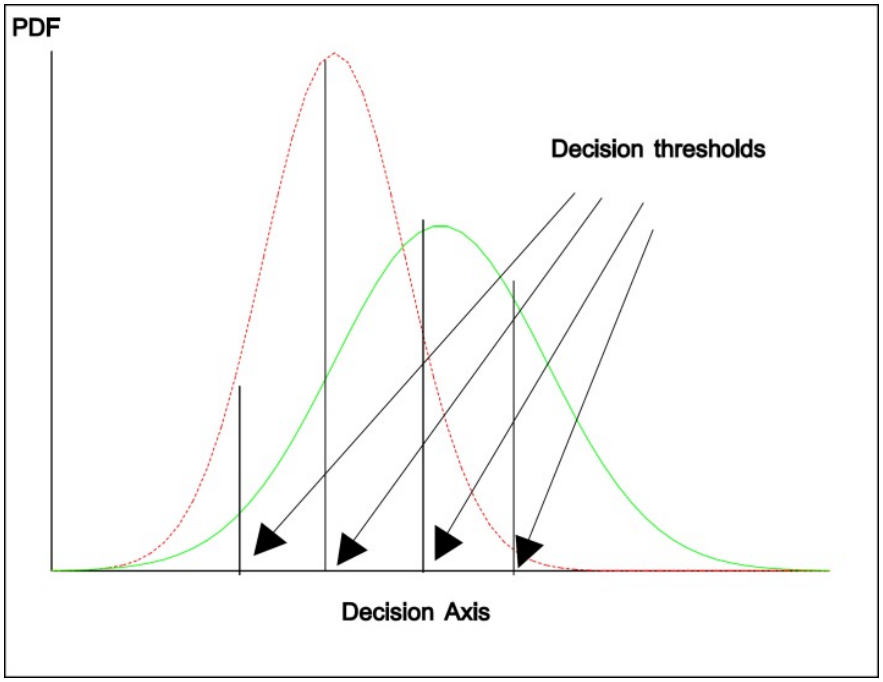

Figure 2.4: Two overlapping probability density functions

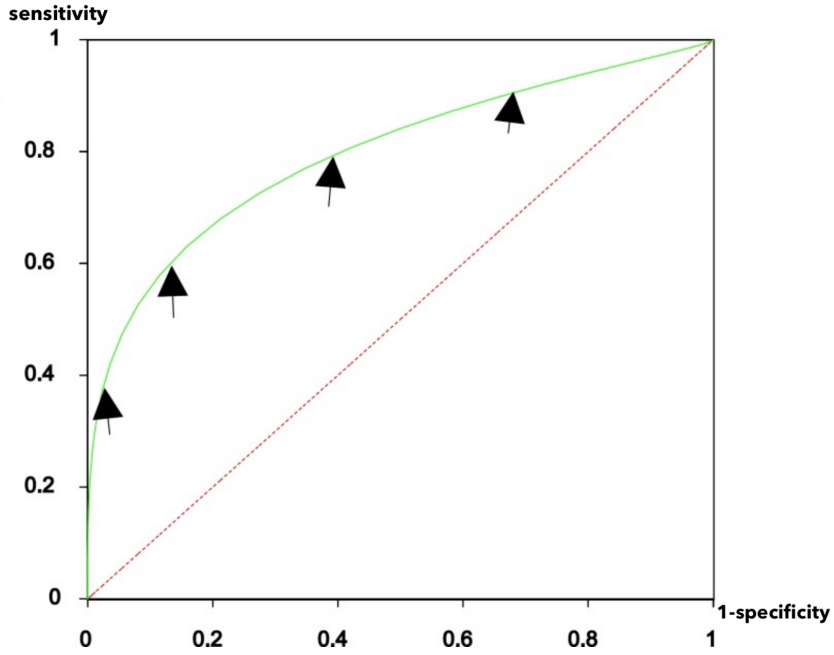

Figure 2.5: ROC curve derived from two overlapping distributions

In the ROC curve, the threshold is located in the upper right when the test is very sensitive but poorly specific. The perfect model is at the top left corner when sensitivity and specificity are both equal to 1, and this is the case when there is no overlapping of two distributions. Thus, two classes can be concluded perfectly discriminated. From the ROC plot, the model performance can be further quantified by finding the area under the curve (AUROC), which can be used to compare multiple models because of no requirement of the threshold.

NP-ROC bands

When evaluating models built with NP classification umbrella algorithm, the ROC curve has a shortcoming in identifying the classifiers which satisfy the prespecified type I error upper bound α with a high probability. Tong at al. (2018) proposed NP-ROC bands as a new visualization tool, whose horizontal axis is the type I error upper bound α, and the vertical axis is (1 − type II error conditioning on training data)[3]. The conditional type II error of a classifier is the type II error conditioning on training data.

Precision-Recall curve

The Precision-Recall (PR) curve visualizes the trade-off between precision and recall under different thresholds, which gives more information regarding the model performance than the ROC curve on imbalanced datasets[22]. The precision is plotted on the y-axis, and the recall is on the x-axis, and the area under the PR (AUPRC) curve serves as a summary statistic for comparing multiple models. Mean average precision, or AP, which is an approximation of the mean AUPRC[23]. We use 'AP' to approximate the mean AUPRC in our study.

CHAPTER 3

Heart Disease Prediction

3.1 Data

The heart disease dataset is publicly available on the Kaggle website, which is based on an ongoing cardiovascular study on residents of the town of Framingham, Massachusetts. The dataset contains the patients' information, including demographic, behavioral, and medical risk factors, including over 4,000 records and 15 features. This is a typical imbalanced binary classification problem since 644 patients are classified as having a 10-year risk of CHD (class 0), while 3596 patients are classified otherwise (class 1). We define the class 0 to be the minority class ('having a 10-year risk of CHD'). After rows that contain NAs are removed, the percentage of minority class is about 15%. The objective of this study is to predict whether the patient has a 10-year risk of future coronary heart disease (CHD).

3.2 Methods

Oversampling

The data is firstly split into the training and testing sets with test size equal to 0.2, and then five oversampling methods are employed to the minority class on training data: regular SMOTE, Borderline-SMOTE1, Borderline-SMOTE2, SVM-SMOTE, and Kmeans-SMOTE. For the parameter in SMOTE methods, $k = 5$ is chosen as the number of nearest neighbors used to construct synthetic instances. In Kmeans-SMOTE, $k = 10$ is used as the number of clusters to be found by Kmeans clustering. In order to compare multiple oversampling

methods, all SMOTE variants are used to oversample the same number of minority instances so that both classes have equal sizes. All models are evaluated using the imbalanced testing set that accounts for 20% of the original dataset.

Two features (BMI and glucose) are selected from the training data (imbalanced and balanced), and their probability density functions are plotted for each class. Figure 3.1 and 3.2 show that resampling the data with SMOTE does not change the distribution of the data.

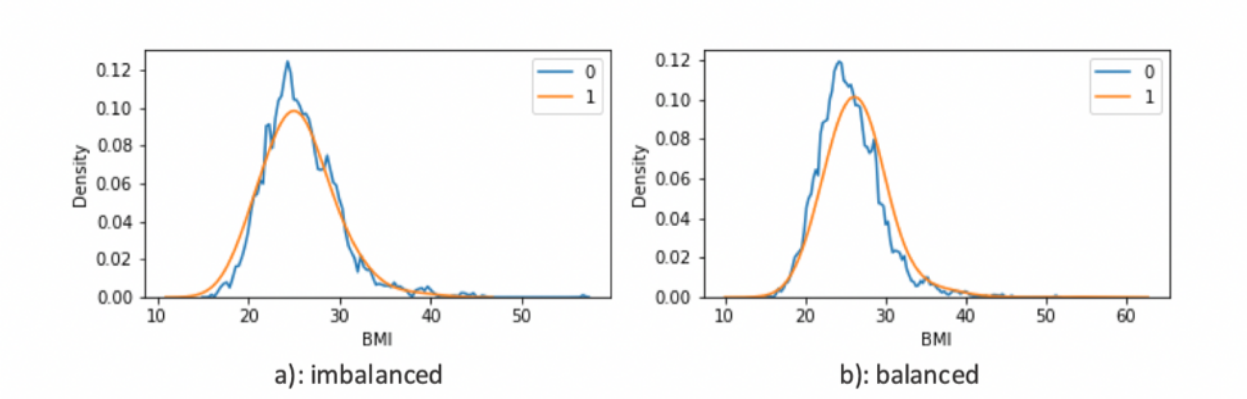

Figure 3.1: The distribution of BMI on the imbalanced and balanced training data

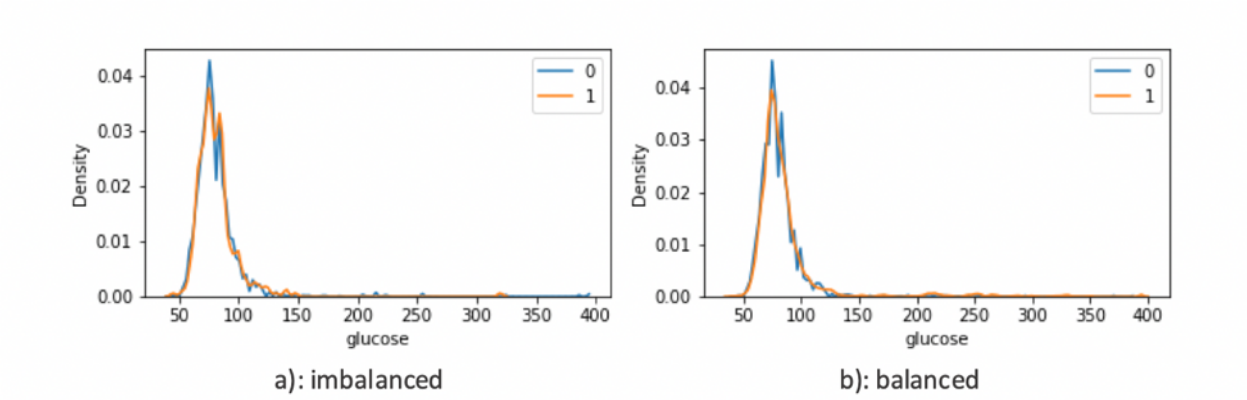

Figure 3.2: The distribution of glucose on the imbalanced and balanced training data

Classification algorithms

Four classical classification algorithms, logistic regression with regularization (L2), random forest, adaboost, and SVM, are applied to build predictive models. In the training process, 5-fold cross-validations are performed to search for optimal hyperparameters. The classifiers with the best parameters are fitted into the training data to obtain a model and then test on the remaining 20% data. The performance of these models is compared with the following criteria: sensitivity, specificity, accuracy, F1-score, AUROC, and AP.

For the NP classification umbrella algorithm, logistic regression and random forest are used as base classification algorithms. The NP classifiers are built with the default value of $\alpha = 0.05$ as the targeted significance level, and $\delta = 0.05$ as the violation rate. For the primary goal of the umbrella algorithm, the type I error and type II error are calculated to evaluate the model performance.

3.3 Results

ROC and PR curves

Figure 3.3 through Figure 3.10 show the ROC and PR curves obtained from SMOTE variants using classical classification algorithms. For each classifier, the curve from the imbalanced data is compared with the curve from the balanced data using SMOTE variants, labeled as "SMOTE", "Borderline-SMOTE1", "Borderline-SMOTE2", "SVM-SMOTE", and "Kmeans-SMOTE". In each plot, we only show three curves with the top three highest AUROC. For example, in Figure 3.3, three curves are from logistic regression (LR) with imbalanced data (labeled as 'LR'), LR with SMOTE, and LR with Borderline-SMOTE1, because they achieve the best three AUROC, followed by the corresponding PR curve plot of the same classifier (Figure 3.4). In the PR curve plot, 'AP' denotes the average mean precision of a classifier.

We can also see that SMOTE consistently performs well since it achieves the best three

AUROC under each classifier (appears in each plot). For logistic regression and random forest, models built with imbalanced data result in similar AUROC values compared with models built with balanced data. Thus there is not enough evidence to indicate the improvement of using SMOTE variants for the AUROC.

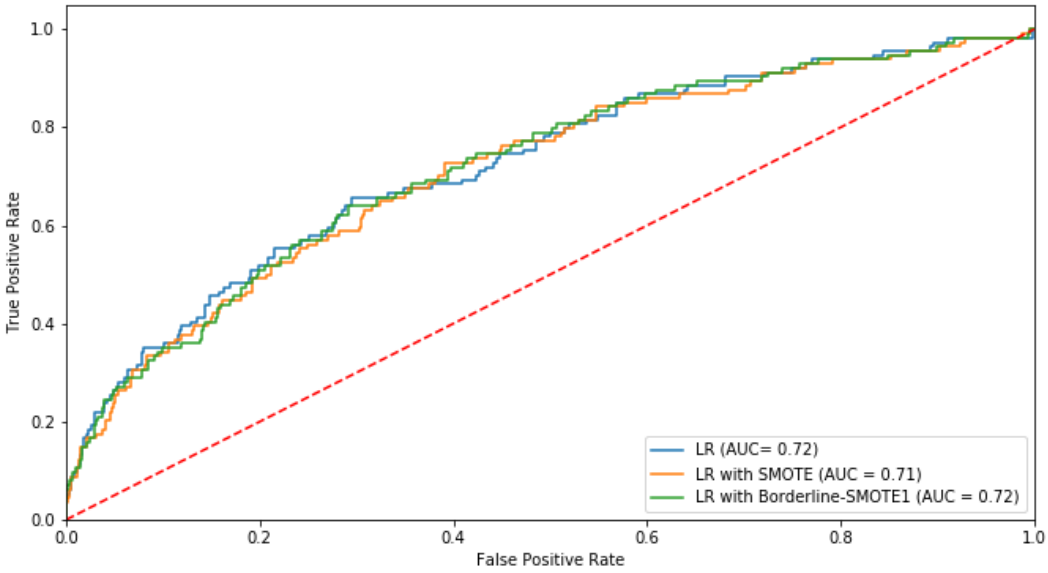

Figure 3.3: Logistic regression ROC curves

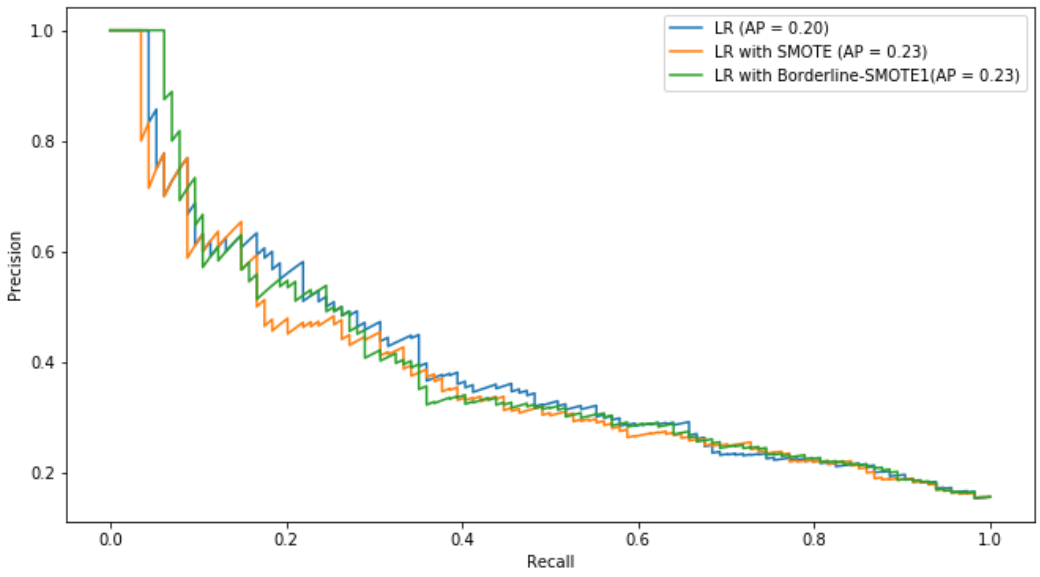

Figure 3.4: Logistic regression PR curves

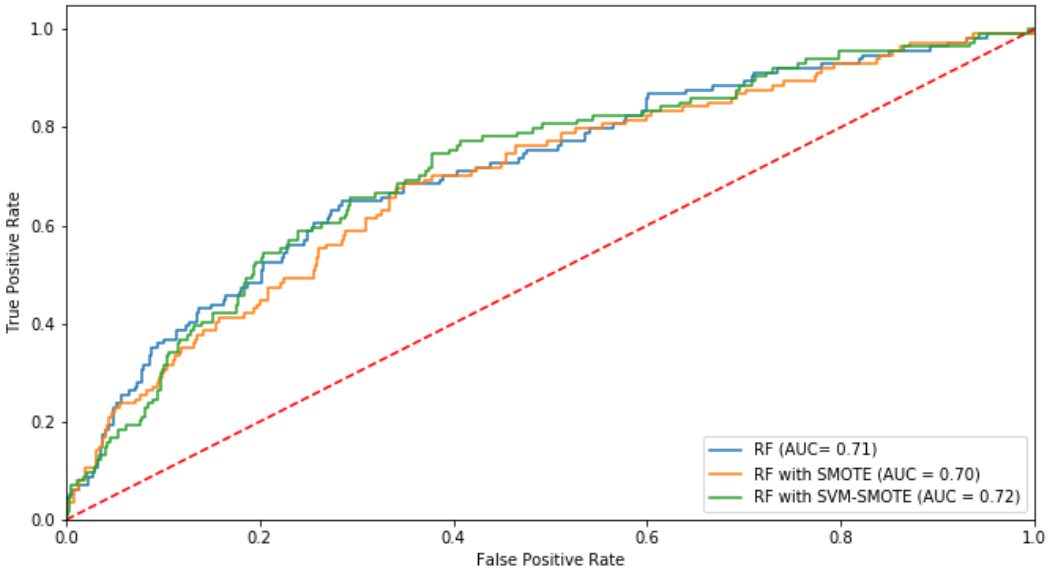

Figure 3.5: Random forest ROC curves

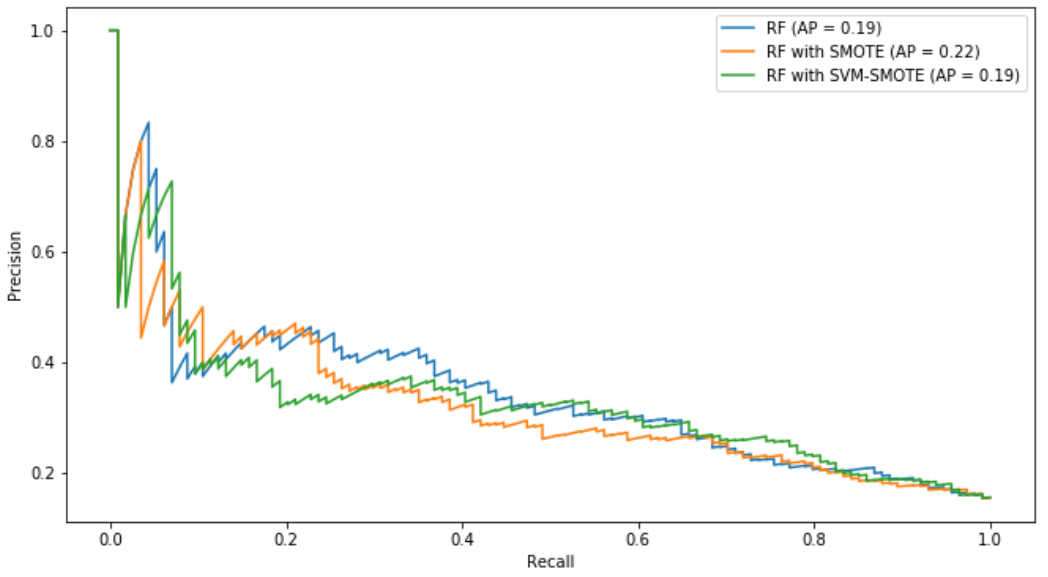

Figure 3.6: Random forest PR curves

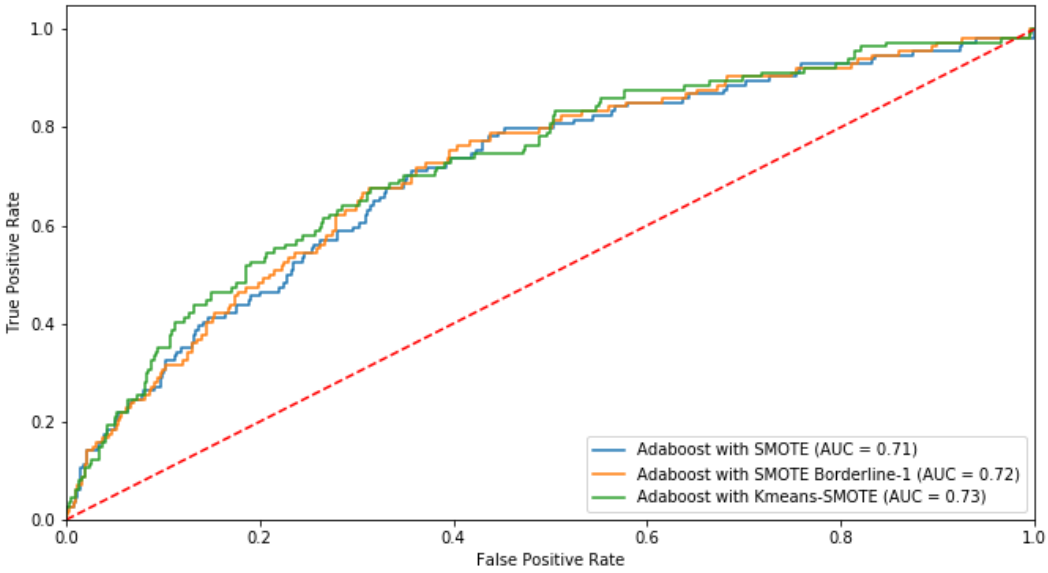

Figure 3.7: Adaboost ROC curves

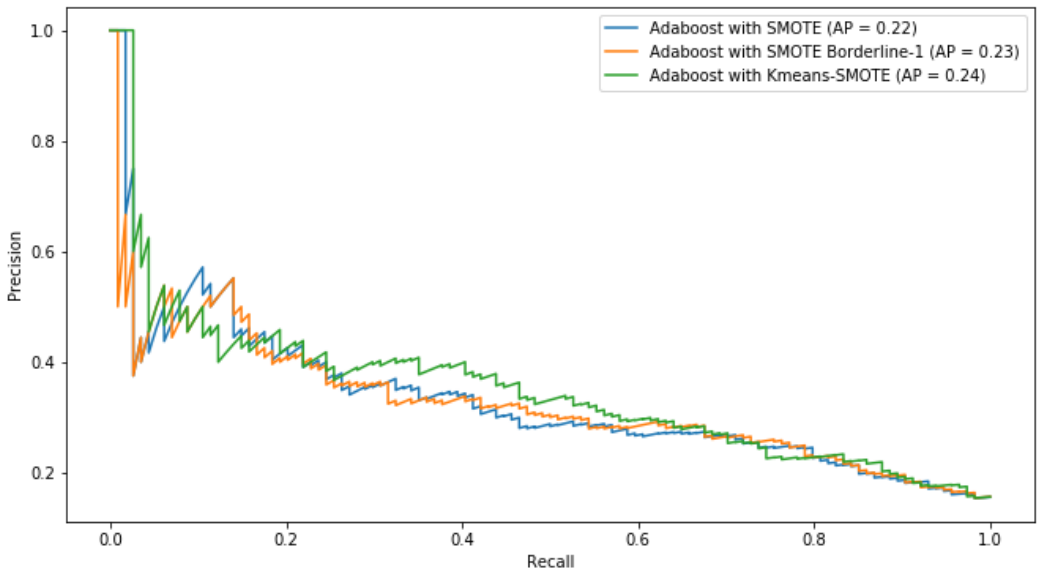

Figure 3.8: Adaboost PR curves

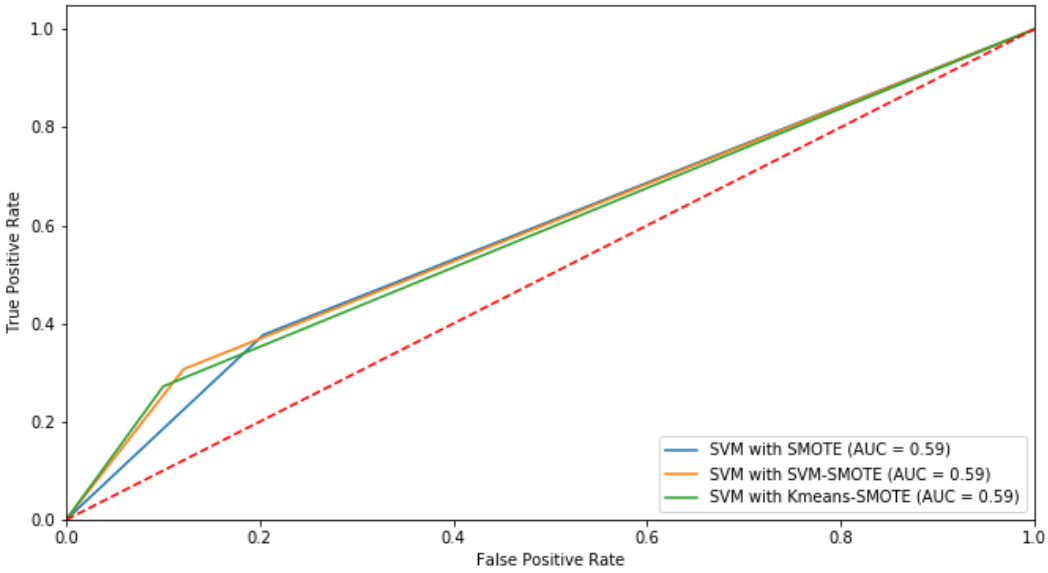

Figure 3.9: Support vector machine ROC curves

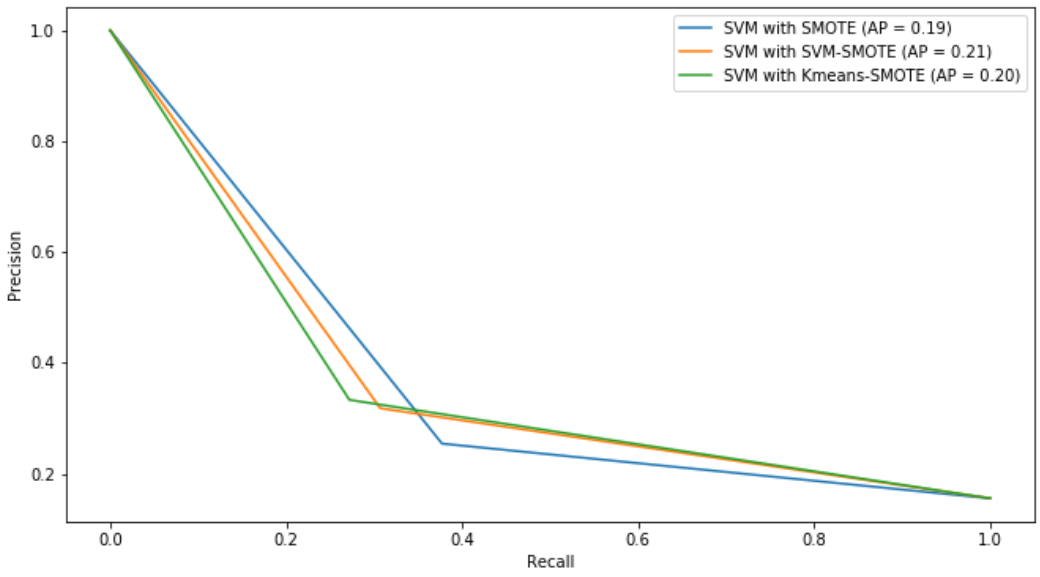

Figure 3.10: Support vector machine PR curves

NP-ROC bands

The following NP-ROC bands are used to compare the performance of NP base classifiers in the NP classification training process. For the imbalanced training data (Figure 3.11), logistic regression performs slightly better in the mean of AUROC. For the balanced training data, random forest outperforms logistic regression in the mean of AUROC (Figure 3.12). We further investigate the model prediction performance in Table 3.6.

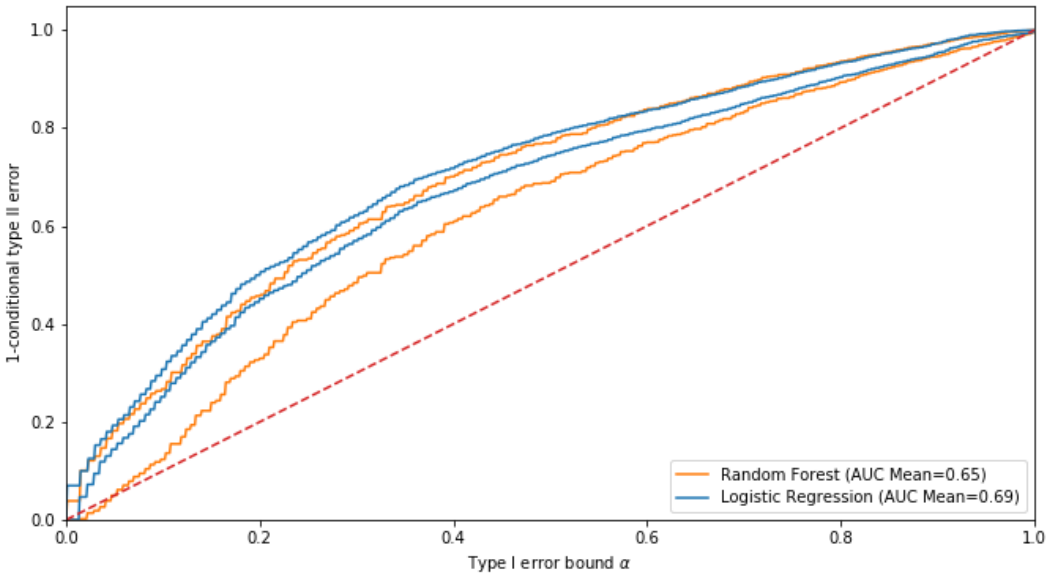

Figure 3.11: NP-ROC bands on the imbalanced (original) training data

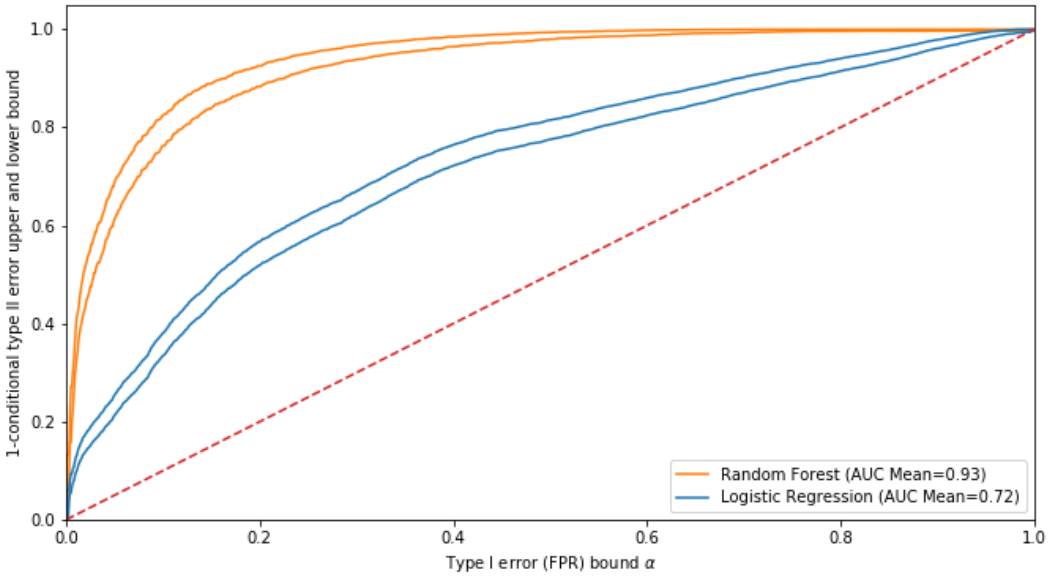

Figure 3.12: NP-ROC bands on the balanced (SMOTE) training data

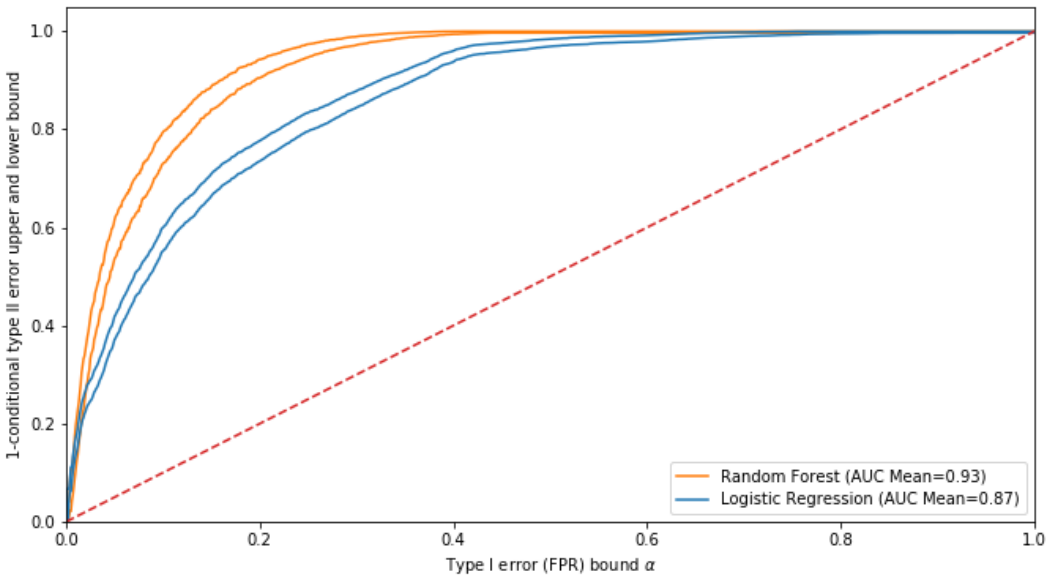

Figure 3.13: NP-ROC bands on the balanced (SVM-SMOTE) training data

Summary statistics for classical classification algorithms

The summary statistics for specificity, sensitivity, accuracy, F1-score, AUROC, and AP for each classification algorithm are listed in Table 3.1 to 3.5. In general, using SMOTE results in higher value of F1-score and AP, lower value of accuracy, and there is no obvious improvement on AUROC criterion. Comparing SMOTE variants, SVM-SMOTE outperforms other SMOTE variants as it results in the best F1-score, AUROC, and AP value in multiple classification algorithms. This result is convincing because SVM-SMOTE not only overcomes the limitations of the regular SMOTE by generating synthetic instances near the borderline of the minority class but also expands the minority region using the extrapolation technique comparing with Borderline-SMOTE.

For random forest, SVM, and adaboost, using SMOTE variants can improve the model performance, because F1-score, AUROC, and AP values all increase compared with that in imbalanced data. However, for logistic regression, there is not enough evidence to show the improvement of using SMOTE variants, except that the F1-score has an average 0.2 increase on the balanced data with logistic regression.

Table 3.5 lists the best method under each evaluation criterion. Borderline-SMOTE2 and SVM-SMOTE are the two best choices of SMOTE variants. Borderline-SMOTE2 with Adaboost achieves the highest value in terms of specificity and sensitivity. SVM-SMOTE with logistic regression gives the best F1-score and AP value.

Table 3.1: Summary statistics for logistic regression with SMOTE variants

Method	Specificity	Sensitivity	Accuracy	F1-score	AUROC	AP
original	**0.6892**	**0.6868**	**0.8511**	0.1550	**0.7222**	0.2006
SMOTE	0.6680	0.6792	0.6503	0.3756	0.7141	0.2263
Borderline1-SMOTE	0.6698	0.6844	0.6749	0.3802	0.7202	0.2291
Borderline2˙SMOTE	0.6627	0.6836	0.6530	0.3682	0.7194	0.2214
SVM-SMOTE	0.6800	0.6825	0.7759	**0.4143**	0.7166	**0.2543**
Kmeans-SMOTE	0.6745	0.6821	0.7281	0.3988	0.7160	0.2417

Table 3.2: Summary statistics for random forest with SMOTE variants

Method	Specificity	Sensitivity	Accuracy	F1-score	AUROC	AP
original	0.6674	0.6558	**0.8470**	0.1515	0.6869	0.1908
SMOTE	0.6553	0.6748	0.7268	0.3630	0.7091	0.2204
Borderline1-SMOTE	0.6824	0.6800	0.7240	0.3804	0.7154	0.2301
Borderline2-SMOTE	0.6489	0.6900	0.8101	0.3594	0.7230	0.2320
SVM-SMOTE	**0.6837**	**0.6789**	0.8087	0.3966	**0.7127**	0.2502
Kmeans-SMOTE	0.6815	0.6895	0.7651	**0.3986**	0.7259	**0.2436**

Table 3.3: Summary statistics for support vector machine with SMOTE variants

Method	Specificity	Sensitivity	Accuracy	F1-score	AUROC	AP
original	**0.6580**	0.3801	**0.8443**	0.2192	0.5572	0.2041
SMOTE	0.5987	0.4591	0.7309	0.3039	0.5867	0.1930
Borderline1-SMOTE	0.6067	0.4474	0.7459	0.2955	0.5812	0.1914
Borderline2-SMOTE	0.6160	0.4357	0.7637	0.2881	0.5775	0.1912
SVM-SMOTE	0.6262	**0.4357**	0.7896	0.3125	**0.5928**	0.2056
Kmeans-SMOTE	0.6332	0.4240	0.8019	**0.2995**	0.5858	**0.2040**

Table 3.4: Summary statistics for adaboost with SMOTE variants

Method	Specificity	Sensitivity	Accuracy	F1-score	AUROC	AP
original	0.6528	0.6558	**0.8470**	0.1515	0.5374	0.1908
SMOTE	0.6562	0.6748	0.7268	0.3631	0.6243	0.2204
Borderline1-SMOTE	0.6923	0.6800	0.7240	0.3804	0.6506	0.2301
Borderline2-SMOTE	**0.6954**	**0.6897**	0.8101	0.3594	0.6193	0.2320
SVM-SMOTE	0.6748	0.6789	0.8087	0.3966	0.6435	**0.2502**
Kmeans-SMOTE	0.6840	0.6895	0.7650	**0.3986**	**0.6570**	0.2436

*Note: the number in boldface is the highest value achieved from the given method.

Table 3.5: The best method under each evaluation criterion

Evaluation	Method	Values
Specificity	Adaboost + BorderlineSMOTE2	0.6954
Sensitivity	Adaboost + BorderlineSMOTE2	0.6897
Accuracy	Logistic Regression	0.8511
F1-score	Logistic Regression + SVM-SMOTE	0.4143
AUROC	Logistic Regression	0.7222
AP	Logistic Regression + SVM-SMOTE	0.2543

Comparison between classical and NP classification algorithms

Table 3.6 lists the type I errors and type II errors from classical and NP classification (labeled bold) algorithms. For simplicity, only regular SMOTE and SVM-SMOTE are used for oversampling the minority class, and logistic regression and random forest are trained as base classification algorithms for comparison. The table shows that NP classification umbrella algorithm effectively controls the type I error under α level 0.05, and it can deal with both imbalanced and balanced data. We also see the cost of lowering the type I error is the increase of the type II error.

Table 3.6: Comparison between classical and NP classification algorithms

Classifier/NP	Method	Type I Error (1-Specificity)	Type II Error (1-Sensitivity)
LR	original	0.3108	0.3132
RF	original	0.3326	0.3442
NP+LR	**original**	**0.0439**	**0.8641**
NP+RF	**original**	**0.0088**	**0.9725**
LR	SMOTE	0.3320	0.3208
RF	SMOTE	0.3447	0.3252
LR	SVM-SMOTE	0.4013	0.5409
RF	SVM-SMOTE	0.3163	0.3211
NP+LR	**SMOTE**	**0.0439**	**0.8754**
NP+RF	**SMOTE**	**0.0175**	**0.9693**
NP+LR	**SVM-SMOTE**	**0.0426**	**0.8301**
NP+RF	**SVM-SMOTE**	**0.0011**	**0.9935**

CHAPTER 4

Conclusion

In this study, SMOTE variants are implemented as oversampling techniques, which are combined with multiple classification algorithms for prediction. By examining the model performance under multiple criteria, SMOTE variants are found not to yield the expected results consistently. From the analysis of the model performance, we can conclude the following:

- SVM-SMOTE and Borderline-SMOTE outperform other SMOTE variants, showing the importance of the borderline instances of the minority class.
- Comparing classical classification algorithms, logistic regression and random forest perform better than SVM and adaboost under multiple evaluation criteria.
- The NP classification umbrella algorithm controls the type I error effectively on both imbalanced and balanced data. In general, random forest as a base classification algorithm results in lower type I error than logistic regression.

In summary, SMOTE variants indicate that the improvement of model performance depends on the evaluation criteria used by researchers. There is no perfect way to report how accurate a test is in clinical settings, as each criterion has its drawbacks. It is important to understand the objective of a specific test. For example, if controlling the type I error is the primary goal of a study, then the NP classification umbrella algorithm is a better choice.

There are several topics left to be considered for future studies. (1) Utilizing different algorithms to identify borderline instances of the minority class would be valuable. For

example, we can apply different grouping strategies to categorize the minority class into different groups and oversample one or two groups. (2) When applying the Kmeans clustering algorithm in Kmeans-SMOTE, finding optimal values of k and other hyperparameters is yet to be guided by rules of thumb. We can study the relationship between optimal hyperparameters for a given dataset and how those parameters are related to the properties of the data.

REFERENCES

[1] Michael Pazzani, Christopher Merz, Patrick Murphy, Kamal Ali, Timothy Hume, and Clifford Brunk. Reducing misclassification costs. pages 217–225, 1994.

[2] Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. pages 155–164, 1999.

[3] Xin Tong, Yang Feng, and Jingyi Jessica Li. Neyman-pearson classification algorithms and np receiver operating characteristics. *Science advances*, 4(2):eaao1659, 2018.

[4] Nicola Torelli and Giovanna Menardi. Training and assessing classification rules with unbalanced data. *Data Mining and Knowledge Discovery*, 2011.

[5] John A Swets. Roc analysis applied to the evaluation of medical imaging techniques. *Investigative radiology*, 14(2):109–121, 1979.

[6] Kendrick Boyd, Vitor Santos Costa, Jesse Davis, and C David Page. Unachievable region in precision-recall space and its effect on empirical evaluation. *ICML'13: Proceedings of the 30th International Conference on International Conference on Machine Learning*, 28:349, 2012.

[7] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[8] Thien M Ha and Horst Bunke. Off-line, handwritten numeral recognition by perturbation method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):535–539, 1997.

[9] Bogdan Gulowaty and Paweł Ksieniewicz. Smote algorithm variations in balancing data streams. pages 305–312, 2019.

[10] Lara Lusa et al. Smote for high-dimensional class-imbalanced data. *BMC bioinformatics*, 14(1):106, 2013.

[11] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. pages 878–887, 2005.

[12] Agostino Cortesi and Nabendu Chaki. Computer information systems and industrial management. 2012.

[13] Hien M Nguyen, Eric W Cooper, and Katsuari Kamei. Borderline over-sampling for imbalanced data classification. *International Journal of Knowledge Engineering and Soft Data Paradigms*, 3(1):4–21, 2011.

[14] Felix Last, Georgios Douzas, and Fernando Bacao. Oversampling for imbalanced learning based on k-means and smote. *arXiv:1711.00837v2*, 2017.

[15] David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

[16] David I Warton. Penalized normal likelihood and ridge regularization of correlation and covariance matrices. *Journal of the American Statistical Association*, 103(481):340–349, 2008.

[17] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of computational and graphical statistics*, 22(2):231–245, 2013.

[18] Christine De Mol, Ernesto De Vito, and Lorenzo Rosasco. Elastic-net regularization in learning theory. *Journal of Complexity*, 25(2):201–230, 2009.

[19] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[20] Robert E Schapire. Explaining adaboost. In *Empirical inference*, pages 37–52. Springer, 2013.

[21] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[22] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3), 2015.

[23] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. 2008.