UCLA
UCLA Electronic Theses and Dissertations

Title
Neyman-Pearson Classification for Credit Card Fraud Detection

Permalink
https://escholarship.org/uc/item/0h39x88k

Author
Xu, Lu

Publication Date
2020

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA
Los Angeles

Neyman-Pearson Classification for Credit Card Fraud Detection

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

by

Lu Xu

2020

© Copyright by
Lu Xu
2020

ABSTRACT OF THE THESIS

Neyman-Pearson Classification
for Credit Card Fraud Detection

by

Lu Xu
Master of Science in Statistics
University of California, Los Angeles, 2020
Professor Jingyi Li, Chair

Fraud detection is a problem of highly imbalanced binary classification, where the size ratio of positive class (fraudulent instances) to negative class (normal instances) is low. Any fraud detection model is faced with the trade-off between two types of classification mistakes - type I error (also known as the false positive rate (FPR)) and type II error (also known as the false negative rate (FNR)). One common way to handle this trade-off is to set a pre-determined level of maximum acceptable type I error, α, and try to minimize type II error. However, an acceptable empirical type I error evaluated on the test set (below α) does not necessarily guarantee population type I error to be below α as well. This problem is addressed by the Neyman-Pearson (NP) classification algorithm (Tong et al. 2018), with which we can put a high-probability upper bound on population type I error. In this study, we use simulation to illustrate how empirical and population type I error can differ, and how NP classification effectively controls population type I error. We then build credit card fraud prediction models based on classical and NP algorithms and compare their performances.

The thesis of Lu Xu is approved.

Hongquan Xu

Guido Fra Montufar Cuartas

Jingyi Li, Committee Chair

University of California, Los Angeles

2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

Fraud detection for credit cards is important for card issuers and large businesses with payment systems. When fraudsters use stolen cards to make purchases, issuers, or business owners, have to bear the loss when true cardholders file chargebacks for the fraudulent transactions to get their money back. Large corporations such as Facebook and Airbnb absorb all the loss when chargebacks for transactions on their platforms take place. These businesses face a trade-off between false positive and false negative when building and implementing their classification models. A false positive case (type I error) occurs when a positive data point is predicted as negative by the classification model, i.e. when a normal transaction/card/customer is predicted as fraudulent. A false negative case (type II error) occurs when a negative data point is predicted as positive, i.e. when a fraudster is predicted as normal. If false positive rate is high, businesses lose some potential revenue from normal cardholders being blocked (or undergoing other friction such as extra authentication). If false negative rate is high, businesses lose the amount of the chargebacks and face the risk of entering compliance programs with card issuers. More often than not, fraud detection is a highly skewed binary classification problem, with one class (the fraudulent class) typically making up less than 5% of the whole population.

It is common to use receiver operating characteristic (ROC) curves to visualize and measure the performance of a binary classification model, and to decide which model and what threshold to use. However, what most practitioners do not realize is that empirical ROC curves based on empirical errors in test set usually differ from ROC curves measured by population error, as pointed out by Tong et al. (2018). For example, if a payment team had

set the goal to limit the type I error below 5%, built models and chosen the one with the lowest empirical type II error when empirical type I error is 5%, the actual type I error after the model is implemented might go beyond 5%.

Precision-recall (PR) curve is another major tool to evaluate a classification model. Unlike ROC curves, PR curves are sensitive to class imbalance. Saito and Rehmsmeier (2015) argues that PR curves are more informative than ROCs when it comes to imbalanced data. The aforementioned problem of discrepancy between empirical and population measures applies to the use of PR curves as well.

The discrepancy between empirical v.s. population type I error is not easily noticed because the true population type I error is usually not traceable. When we take enforcement on cases labeled as positive to block their future actions, we do not know whether it is a true negative or a false positive in reality. In the credit card fraud context, when a card is predicted as fraudulent and blocked, we are not able to tell if it was going to make a normal purchase or a bad purchase resulting in chargebacks. Thus, it is imperative to have a scientific method that has more control over the population type I error.

Tong et al. (2018) proposes the NP classification algorithm which puts a high-probability upper bound on population type I error while minimizing type II error. The NP algorithm is an umbrella algorithm that builds on scoring-type base algorithms such as penalized logistic regression (penLR), random forest (RF) and support vector machine (SVM). The basic idea of this umbrella algorithm is to re-select thresholds for the base algorithm to limit type I error under a desirable level with high probability. A variant of the ROC curve, the NP-ROC interval, is introduced as a graphical evaluation measure for the NP classifiers. On the NP-ROC plot, we can see the high-probability upper and lower bounds of the conditional type II error for each level of high-probability upper bound for type I error (α), so that we can choose a most suitable α based on business needs.

In this study, we will first use simulated balanced and imbalanced data sets to illustrate the aforementioned discrepancy in empirical and population measures. We will see how NP classifiers effectively control the type I error below our desired level. We will then train classical and NP classifiers for a real credit card fraud data set to compare the performance between these classifiers. In addition, we will illustrate how to use an NP-ROC plot to choose a desirable type I error upper bound.

CHAPTER 2

Methods

2.1 Binary classification with imbalanced data

2.1.1 Classical classification algorithms

Suppose we have p predictors $X = (X_1, ..., X_p)$ and a binary response variable $Y \in \{0, 1\}$.

Logistic regression

Logistic regression is the most basic statistical model used for binary classification. It makes use of the logistic function $p(X) = \mathbb{P}(Y = 1|X) = \frac{\exp(\beta_0+\beta_1 X_1+...+\beta_p X_p)}{1+\exp(\beta_0+\beta_1 X_1+...+\beta_p X_p)}$ to get the linear relationship $\log(\frac{p(X)}{1-p(X)}) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p$. We usually use maximum likelihood estimation to fit the logistic regression model and estimate the coefficients. We choose the estimates $\hat{\beta}$ that maximize the likelihood function $l(\beta) = \prod_{i:y_i=1} p(x_i) \prod_{j:y_j=0}(1 - p(x_j))$, or equivalently the log-likelihood function $\sum_{i:y_i=1} \log(p(x_i)) + \sum_{j:y_j=0} \log(1 - p(x_j))$.

From the log-likelihood function, we can tell that there would be a problem when fitting the model with highly skewed data (without any over-sampling or under-sampling of instances). The minority class has a much smaller influence on the objective function than the majority class does.

Penalized logistic regression

Penalized logistic regression model adds a penalty (regularization) term to the above logistic model to prevent the model from becoming too complex. Common regularization methods include ridge, lasso, and elastic net regression.

Decision tree

Decision trees segment a data set into smaller and smaller subsets according to certain split rules. A tree can be represented by a graph with internal decision nodes and terminal nodes (leaves). At each internal decision node, the instances are split in a way that increases the homogeneity the most. Possible split criteria include Gini impurity, information gain and information gain ratio. The splitting is stopped when each leaf node has fewer than some minimum number of instances. It is common to grow a full tree and then prune it to reduce the model complexity. After obtaining the final tree, we predict the class of an instance based on the most common class of its corresponding leaf node.

Random forest

Random forest is a tree-based method that utilizes bagging (bootstrap aggregating) of decorrelated trees. Bagging is a process in which the training set is sampled with replacement to fit individual decision trees and final prediction is based on majority vote from all classification trees. Random forests improve the general bagging technique in that a random subset of features is used at each split when training the individual trees, which further decreases the correlation between trees.

2.1.2 Neyman-Pearson (NP) classification umbrella algorithm

In its essence, the NP umbrella algorithm Tong et al. (2018) provides a way to find optimal score thresholds for base scoring-type classifiers, so that the resulting classifier has the nice property of limiting type I error below α with a high probability larger than 1-δ, where α (type I error upper bound) and δ (tolerance level) are predetermined.

The NP umbrella algorithm divides the class 0 instances in the training set into two halves, each of size n. The first half, together with all the class 1 data points, are used for the training of the base classifier (denoted by $f(\cdot)$). The second half is used as a left-out sample to determine the score threshold. Specifically, we apply the base classifier to these n data points to get their scores, denoted by $T_1, ..., T_n$, i.e. $f(x_i) = T_i$ for $i = 1, ..., n$, where x_i is the data

points to be classified. Then, we rank them by increasing order $T_{(1)} \leq ... \leq T_{(n)}$ and construct a classifier using the k^*th order statistic by $\hat{\phi}_{k^*}(x) = \mathbb{1}(f(x) > T_{(k^*)})$, where k^* is the predetermined rank threshold. k^* is obtained by the formula $k^* = \min\{k \in \{1, ..., n\} : v(k) \leq \delta\}$, where $v(k) = \sum_{j=k}^{n} \binom{n}{j}(1-\alpha)^j \alpha^{n-j}$ is the upper bound of the probability that the type I error of $\hat{\phi}_{k^*}$ exceeds α.

In summary, the NP umbrella algorithm consists of three main steps.

- Step 1. Given n, α and δ, calculate the rank threshold k^* to be used.
- Step 2. Randomly split class 0 (in the training set) into two halves. Train a base scoring-type classifier using the first half and class 1. Score the second half and use the k^*th order statistic as the threshold for the base classifier.
- Step 3. Repeat Step 2 M times to get an ensemble NP classifier by majority vote.

2.2 Under- and over-sampling

When standard classification algorithms built with the goal of reducing misclassification error are applied on imbalanced data, the results can be misleading as the minority class is under-represented during training (Amin et al. 2016). Thus, with imbalanced data, we typically apply sampling to obtain a data set with balanced class before model fitting. The most direct way to achieve this goal is random under-sampling of majority class or over-sampling of the minority class. However, it is possible that random under-sampling causes information loss about the majority class, and random over-sampling leads to repeated minority instances resulting in overfitting (He and Garcia 2009). More advanced sampling methods are developed and we summarize some of them below.

2.2.1 NearMiss

NearMiss algorithm (Zhang and Mani 2003) is an undersampling technique that aims to mitigate the information loss during undersampling of the majority class. There are three methods to select near-miss instances.

- NearMiss-1. Select majority instances who have the smallest average distances to their nearest k minority neighbors, where k is a user-specified value.
- NearMiss-2. Select majority instances who have the largest average distances to their farthest k minority neighbors, where k is a user-specified value.
- NearMiss-3. Select a given number of the closest majority instances for each minority instance.

2.2.2 Synthetic minority oversampling technique (SMOTE)

Synthetic minority oversampling technique (SMOTE) is a technique where the minority class is over-sampled by creating synthetic instances (Bowyer et al. 2002). Synthetic instances are created such that they fall randomly on the line joining one of the k nearest neighbors of a minority instance and itself. Compared to random oversampling, SMOTE not only increases the size of the minority class, but also the variety of minority instances.

2.3 Evaluation of classifiers

As Japkowicz and Shah (2011) has summarized, the evaluation of learning algorithms is both in absolute and relative terms. The performance measure is a main area in the evaluation, and it contains both summary statistics and graphical measures. We will only discuss the evaluation for binary classification algorithms here.

2.3.1 Confusion matrix and related summary statistics

Confusion matrix

The confusion matrix is a basic performance measure for classifiers, upon which many other more complex measures are built. For the binary classification case, a confusion matrix is a 2-by-2 matrix $\{c_{ij}\}, i, j = 0, 1$, where c_{ij} represents the number of instances from class i that are predicted to be class j by the classifier. Throughout the paper, we refer to the minority class as class 1 or the positive class.

	Predicted negative	Predicted positive
Actual negative (N)	True negative (TN)	False positive (FP)
Actual positive (P)	False negative (FN)	True positive (TP)

Table 2.1: Confusion matrix for binary classification

We see that TN and TP are data points from class 0 and 1 respectively that are classified correctly, while FN and FP are mistakes. Nearly all models make mistakes, and we have to decide how to manage the trade-off between these two mistakes.

Row normalization of the confusion matrix gives us more metrics. The true positive rate and the false negative are defined as a proportion of the positive class.

$$\text{TPR} = \frac{\text{TP}}{\text{P}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$\text{FNR} = \frac{\text{FN}}{\text{P}} = \frac{\text{FN}}{\text{TP} + \text{FN}} = 1 - \text{TPR}$$

Similarly, the true negative rate and the false positive are defined as a proportion of the negative class.

$$\text{TNR} = \frac{\text{TN}}{\text{N}} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$
$$\text{FPR} = \frac{\text{FP}}{\text{N}} = \frac{\text{FP}}{\text{TN} + \text{FP}} = 1 - \text{TNR}$$

Elements in the confusion matrix serve as a building block to construct many other performance measures.

Sensitivity, specificity and likelihood ratio

One aspect of the evaluation of a classifier is to see how many times more likely instances from class 0 (or 1) are labeled class 0 (or 1) than instances from the other class.

Sensitivity of a classifier is defined to be the TPR. It focuses on measuring how many of the positive cases can be successfully identified. In the credit card fraud detection, this is the proportion of fraudsters that are correctly detected.

Specificity is a complementary metric to sensitivity. It is defined to be the TNR. It focuses on measuring how many of the negative cases can be successfully identified. That is, the proportion of normal cardholders that are labeled as normal.

Likelihood ratio aims to combine sensitivity and specificity into one metric.

$$\begin{aligned} \mathrm{LR}_+ &= \frac{\text{Sensitivity}}{1-\text{Specificity}} = \frac{\mathrm{TPR}}{1-\mathrm{TNR}} = \frac{\mathrm{TPR}}{\mathrm{FPR}} \\ &= \frac{\%\text{ of positive instances predicted positive}}{\%\text{ of negative instances predicted positive}} \end{aligned}$$

$$\begin{aligned} \mathrm{LR}_- &= \frac{\text{Specificity}}{1-\text{Sensitivity}} = \frac{\mathrm{TNR}}{1-\mathrm{TPR}} = \frac{\mathrm{TNR}}{\mathrm{FNR}} \\ &= \frac{\%\text{ of positive instances predicted negative}}{\%\text{ of negative instances predicted negative}} \end{aligned}$$

In words, LR_+ measures how many times more likely positive instances are to have a positive prediction than negative class. LR_- measures how many times less likely positive instances are to have a negative prediction than negative class. Higher LR_+ and lower LR_- indicate a better classifier.

Predictive values

Positive predictive value (PPV) and negative predictive value (NPV) aims to measure what proportion of the positive (and negative) predictions are predicted correctly.

$$\begin{aligned} \mathrm{PPV} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} = \frac{\mathrm{TP}}{\#\text{ predicted positive}} \\ \mathrm{NPV} &= \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FN}} = \frac{\mathrm{TN}}{\#\text{ predicted negative}} \end{aligned}$$

We should note that what is different here is that the PPV and NPV are not based on row-normalized elements of the confusion matrix. Thus the balance between the two classes

would have an effect on these metrics.

Figure 2.1 illustrates the confusion matrix in imbalanced binary classification. The illustration shows the case when positive instances make up 25% of all instances and when FPR = FNR = 0.2. Although the FPR and FNR are at the same level, PPV= $\frac{80}{140} = 0.57$ and NPV= $\frac{240}{260} = 0.92$ are very different. This difference comes from the imbalanced nature of the instances.

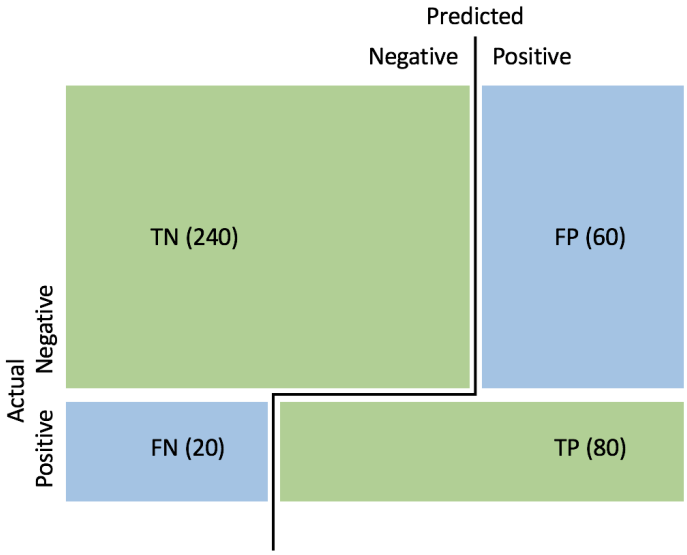


Figure 2.1: Confusion matrix for imbalanced classification

Precision, recall and F measure

In this aspect of evaluation, we focus more on the positive class than the negative class. The precision of a classifier is just the PPV defined above, and recall is just the TPR. From the formulae, we see precision and recall measure the TP as a proportion of predicted positive and actual positive respectively.

$$\text{Precision} = \text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\#\text{ predicted positive}}$$
$$\text{Recall} = \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\#\text{ actual positive}}$$

Again, attempts are made to combine precision and recall into a single metric. The F measure is defined to be a weighted harmonic mean of precision and recall.

$$F_\alpha = \frac{(1+\alpha)(\text{Precision} \times \text{Recall})}{\alpha \times \text{Precision} + \text{Recall}}, \quad \alpha > 0.$$

The weight, α, is decided by the trade-off between the importance of precision and recall. Commonly used F measures include F_1 (balanced importance), F_2 (more importance on recall), and $F_{0.5}$ (more importance on precision).

Common summary statistics

Measure	**Formula**	**Sensitive to class imbalance**
TPR (a.k.a. sensitivity, recall)	$\frac{\text{TP}}{\text{TP+FN}}$	No
FPR	$\frac{\text{FP}}{\text{TN+FP}}$	No
LR_+	$\frac{\text{TPR}}{\text{FPR}}$	No
PPV (a.k.a. precision)	$\frac{\text{TP}}{\text{TP+FP}}$	Yes
F_α measure	$\frac{(1+\alpha)(\text{PPV}\times\text{TPR})}{\alpha\times\text{PPV+TPR}}$	Yes

Table 2.2: A list of some common summary statistics and their relationship to each other.

2.3.2 Classical graphical measures

Here, we discuss graphical analysis tools for evaluation of the performance of scoring classifiers. Scoring classifiers assign scores to each instance and then use a threshold to decide binary labels. Classifiers with the same scoring function but different thresholds would have different confusion matrix. When a threshold is already chosen, the confusion matrix gives us adequate information for performance analysis. However, if the threshold is not yet chosen and we want to analyze the performance of the scoring algorithms over all possible thresholds, graphical measures would be more informative tools to use.

Empirical receiver operating characteristics (ROC) curves

An ROC curve is a plot of TPR against FPR. For the same scoring function, one threshold corresponds to one single point on its ROC curve. Since TPR $\in [0,1]$ and FPR $\in [0,1]$, the space of ROC is $[0,1] \times [0,1]$, i.e. the unit square. For an extremely low threshold, all instances are labeled positive. In this case, TPR = FPR = 1, so a classifier with extremely low threshold corresponds to the upper right corner in the square of ROC space. As the threshold increases, fewer instances are classified as positive, so FPR and TPR both decreases. When the threshold reaches extremely high value, no instance is labeled positive, making TPR = FPR = 0, i.e. the lower left corner of the square. Thus, an ROC curve always connects $(0,0)$ and $(1,1)$. Also, along the diagonal of the ROC space, TPR = FPR. Any point on the diagonal with TPR = FPR = c is just a random classifier that classifies an instance as positive with probability c, irrespective of its actual class. In general, suppose classifier a with scoring function f_1 and threshold t_1 has an coordinate (x_1, y_1) in the ROC space, and classifier b with scoring function f_2 and threshold t_2 has an coordinate (x_2, y_2). Then classifier a is superior than b if $x_1 < x_2$ and $y_1 > y_2$, i.e. lower FPR and higher TPR. Thus, the closer the ROC curve is to the upper left corner of the unit square, the better the scoring classification algorithm is.

We should note that as ROC curves depend only on TPR and FPR measures, they are insensitive to class imbalances. Whether the positive class makes up 50% or 5% of the whole population will not have any effect on the plot of ROC curves.

The way to plot the ROC curve on a single testing set is straight forward - simply increase the threshold and do the calculation. For the plotting of ROC when k-fold cross-validation is used, there are two mainstream methods. The first is to do the calculation separately for the k folds, and then for each level of FPR, calculate the average TPR. The second is to obtain the scores for instances from all the folds and plot a single curve.

Empirical precision-recall (PR) curves

PR curves plot the precision (i.e. PPV) against recall (i.e. TPR) over all possible thresholds. Similar to ROC curves, the space of a PR curve is a $[0, 1] \times [0, 1]$ unit square. When the threshold is extremely low, all instances are classified as positive. We have recall = 1 and precision = $\frac{P}{P+N}$, the natural proportion of the positive instances. When the threshold is extremely high, all instances are classified as negative. We have recall = 0 and precision undefined. A random classifier in the PR space corresponds to the horizontal line $y = \frac{P}{P+N}$, which is sensitive to class balance.

Essentially, the information carried in a confusion matrix has three degrees of freedom (up to a scalar multiple). As Japkowicz and Shah (2011) have pointed out, we can think of the three dimensions as TPR, FPR and class imbalance ratio (referred to as $c_i = \frac{\#\text{ actual positive}}{\#\text{ actual negative}}$). The ROC curves cover the first two dimensions but neglect the information carried by the third dimension c_i. We cannot construct the confusion matrices from a ROC curve, as information about the relative size of classes is lost.

	Predicted negative	Predicted positive
Actual negative (N)	$(1 - \text{FPR}) \times \lambda$	$\text{FPR} \times \lambda$
Actual positive (P)	$(1 - \text{TPR}) \times \lambda \times c_i$	$\text{TPR} \times \lambda \times c_i$

Table 2.3: Confusion matrix has three degree of freedom up to a scalar multiple λ. Here, we choose the three dimensions to be TPR, FPR and c_i as defined above.

For PR curves, the x-axis (recall) gives us the first dimension TPR. Let's investigate the information carried by the y-axis.

$$\begin{aligned}
\text{PPV} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\
&= \frac{\text{TPR} \times \lambda \times c_i}{\text{TPR} \times \lambda \times c_i + \text{FPR} \times \lambda} \\
&= \frac{\text{TPR}}{\text{TPR} \ + \text{FPR} \times \frac{1}{c_i}} \\
\text{FPR} \times \frac{1}{c_i} &= \frac{\text{TPR} \times (1 - \text{PPV})}{\text{PPV}}
\end{aligned}$$

The PR curves give us information about TPR and the ratio of FPR and c_i. Although we are still unable to reconstruct the confusion matrix from PR curves, at least the third dimension c_i is not completely neglected as in ROC curves.

For a given dataset, i.e. when c_i is fixed and known, there is a one-to-one relationship between the ROC and PR curve of a classifier. Saito and Rehmsmeier (2015) have proved that for a fixed number of c_i, a classifier dominates another in the ROC space if and only if it dominates in the PR space.

2.3.3 Graphical measure under NP paradigm

NP-ROC

Tong et al. (2018) proposed the NP-ROC bands as a graphical measure for the NP classifiers. We summarize this measure below, with some notations carried over from section Neyman-Pearson (NP) classification umbrella algorithm.

The horizontal axis of an NP-ROC plot is the population type I error upper bound with high probability. The vertical axis is (1 - type II error conditioning on training data). An NP classifier corresponds to a vertical line segment in NP-ROC (instead of a point as in ROC), with upper and lower ends corresponding to the upper and lower high-probability bounds of (1 - type II error conditioning on training data) for that NP classifier.

To generate an NP-ROC plot, the NP umbrella algorithm is slightly modified.

- Step 1. Choose a tolerance level δ to be used.
- Step 2. Randomly split both class 0 and class 1 into two halves respectively, denoted by S_1^0, S_2^0 from class 0, and S_1^1, S_2^1 from class 1. Train a base scoring-type classifier using S_1^0 and S_1^1.
- Step 3. Use the base scoring-type classifier to obtain scores for the instances in S_2^0 and S_2^1, and rank the scores respectively. We denote the resulting ranked scores as $t_{(1)}^0, ..., t_{(n)}^0$ for class 0 and $T_{(1)}^1, ..., T_{(m)}^1$ for class 1. For $1 \leq k \leq n$, use the kth order statistic of the scores to find the $(1-\delta)$ probability upper bound of the population type I error (x-coordinate, denoted by $\alpha(\hat{\phi}_k)$), as well as the $(1-\delta)$ probability lower and upper bounds of the conditional type II error (y-coordinates, denoted by $\beta_L(\hat{\phi}_k)$ and $\beta_U(\hat{\phi}_k)$). $\alpha(\hat{\phi}_k), \beta_L(\hat{\phi}_k),$ and $\beta_U(\hat{\phi}_k)$ are defined below.
- Step 4. Repeat Step 2 and 3 multiple times and take vertical average, where we take averages separately for the lower bounds and upper bounds for each x-coordinate.

$\alpha(\hat{\phi}_k), \beta_L(\hat{\phi}_k)$, and $\beta_U(\hat{\phi}_k)$ are defined as follows.

$$\alpha(\hat{\phi}_k) = \inf\{\alpha \in [0,1] : \sum_{j=k}^{n} \binom{n}{j} (1-\alpha)^j \alpha^{n-j} \leq \delta\},$$

$$\beta_L(\hat{\phi}_k) = \sup\{\beta \in [0,1] : \sum_{j=r_L}^{m} \binom{m}{j} \beta^j (1-\beta)^{m-j} \leq \delta\},$$

$$\beta_U(\hat{\phi}_k) = \inf\{\beta \in [0,1] : \sum_{j=r_U}^{m} \binom{m}{j} \beta^j (1-\beta)^{m-j} \geq 1-\delta\},$$

$$\text{with } r_L = \max\{r \in \{1,...,m\} : T^1_{(r)} \leq t^0_{(k)}\},$$

$$\text{and } r_U = \min\{r \in \{1,...,m\} : T^1_{(r)} \geq t^0_{(k)}\},$$

with the exception $\beta_U(\hat{\phi}_k) = 1$ when $t^0_{(k)} > T^1_{(m)}$, and $\beta_L(\hat{\phi}_k) = 0$ when $t^0_{(k)} < T^1_{(1)}$.

2.3.4 Choosing a model based on performance measurement

After building a few classification models, we evaluate them based on the aforementioned measures. Since no model is perfect and we are always faced with the trade-off between type I and type II errors. A systematic way to choose the best model to use is imperative.

Under the classical classification paradigm, the most common procedure to choose the best model is to compare their empirical ROC curves based on a hold-out test set. Given a predetermined FPR, we check the ROC curves to see at that empirical FPR, which model has the highest empirical TPR and select that model and corresponding threshold. As we mentioned, this method faces the problem of discrepancy between empirical and population FPR.

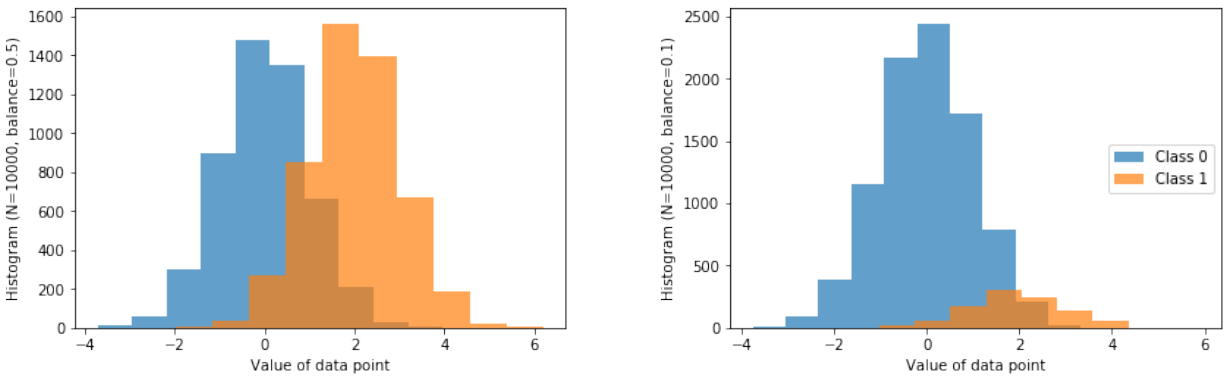
Under the NP classification paradigm, the procedure is different. The predetermined FPR is used as a parameter when constructing the NP classifier. If we are sure about the ideal FPR upper bound to use, we can just calculate the upper and lower high-probability bounds of (1 - conditional type II error) for the classifiers and compare these values. If the ideal FPR is not determined, we can construct the NP-ROC to understand the trade-off and make decisions according to our business needs.

CHAPTER 3

Simulation

3.1 Data

We follow the simulation from Tong et al. (2018), with modification on the class ratio to make the data set possibly imbalanced. The underlying distribution for the two populations are $N(0, 1)$ for class 0 (negative, majority class) and $N(2, 1)$ for class 1 (positive, minority class). There are a total of $10,000$ instances in one data set, with balance defined to be the proportion of class 1 instances in the whole data set (Figure 3.1). We investigate the cases where the two classes are balanced (with balance$= 0.5$) or imbalanced (with balance$= 0.1$).

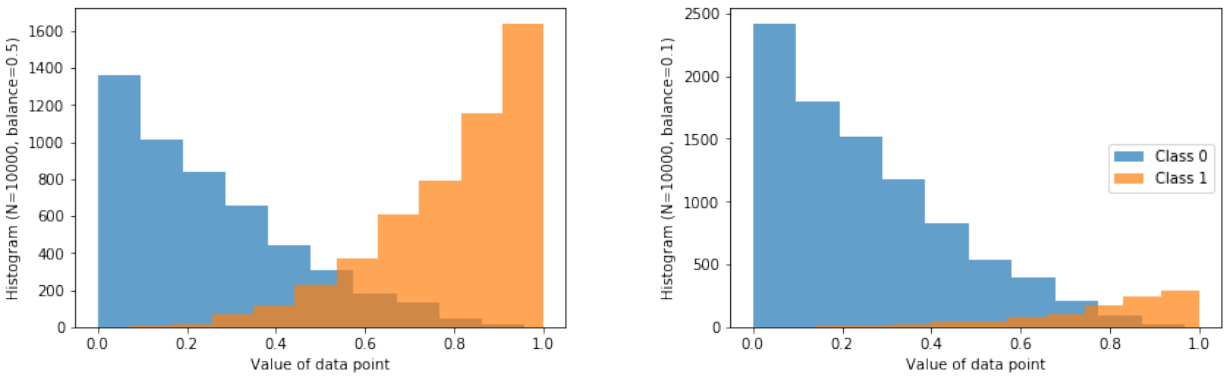


(a) Gaussian, balanced

(b) Guassian, imbalanced

Figure 3.1: Visualization of the balanced and imbalanced Gaussian data sets simulated

In addition, we investigate another simulation with underlying distribution follow Beta$(1, 3)$ for class 0 and Beta$(4, 1)$ for class 1 (Figure 3.2). Size of one data set is $10,000$ as well.



(a) Beta, balanced

(b) Beta, imbalanced

Figure 3.2: Visualization of the balanced and imbalanced Beta distribution data sets

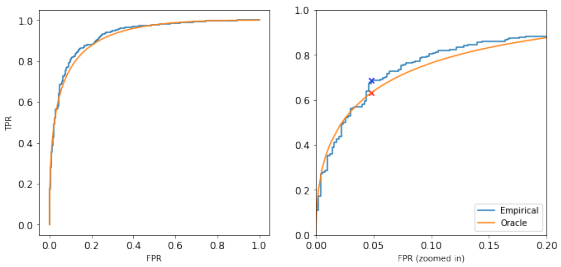
For both simulations, since the feature space is 1-dimensional, the identity function is used as a natural classification scoring function.

3.2 Empirical ROC v.s. oracle (population) ROC curves

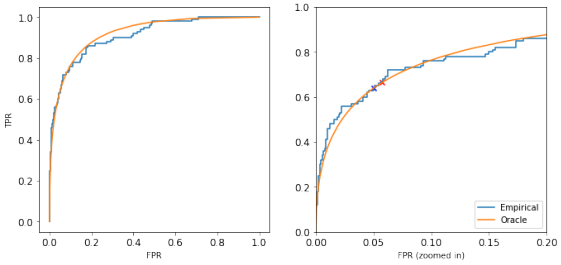
Empirical ROC curves are the most frequently used performance evaluation measure for classification models. Decisions such as which model to use and what threshold to choose are often based on the empirical ROC curves. In this analysis, we aim to illustrate how empirical ROC curves (based on testing data) can be different from the true oracle ROC (based on the whole population), and how true TPR and FPR levels can differ from expectation as a result.

In Figure 3.3, the orange curves are the oracle ROC curve plotted based on the underlying distribution, whereas the blue curves are the empirical ROC curve based on one random dataset of size $10,000$. The blue dots are the level of TPR and FPR we believe we are operating at when choosing the threshold that has an empirical FPR$= 0.05$, while the red dots are the actual population TPR and FPR corresponding to that chosen threshold. The evaluations based on empirical ROC curves can mislead us about the true level of TPR and FPR. It is possible that we are implementing a model that has a higher FPR and lower TPR

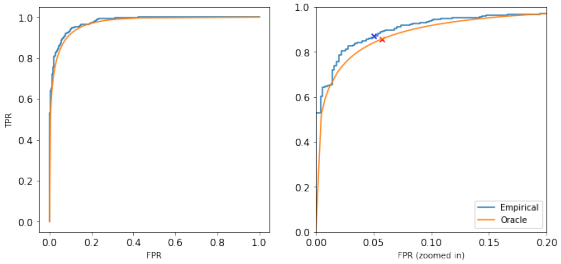
than the level we believe we are at.



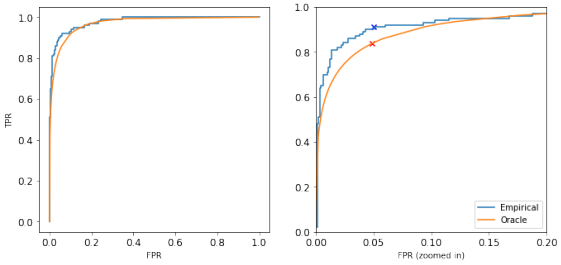
(a) Balanced, Gaussian



(b) Imbalanced, Gaussian

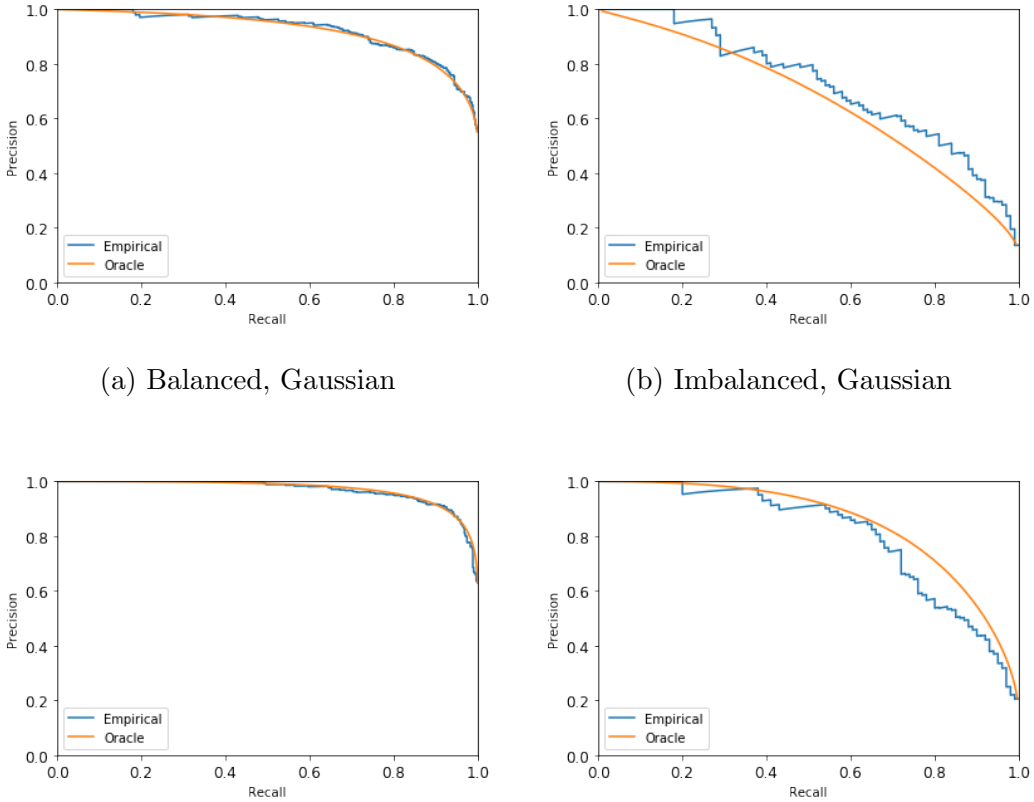


(c) Balanced, Beta



(d) Imbalanced, Beta

Figure 3.3: Empirical v.s. oracle ROC curves



(a) Balanced, Gaussian

(b) Imbalanced, Gaussian

(c) Balanced, Beta

(d) Imbalanced, Beta

Figure 3.4: Empirical v.s. oracle PR curves

Now, we draw $1,000$ random data sets each with $10,000$ data points and perform the empirical v.s. oracle comparison as above. For each data set, we choose the threshold that corresponds to empirical FPR$= 0.05$, and see the variation of oracle TPR and FPR (Figure 3.5). From Table 3.1 and Figure 3.5, we understand that the population FPR and TPR can have a wide range when we choose a model and threshold based on empirical measures evaluated on a hold-out test set. In addition, the desired FPR rate (0.05 in the illustration) is more like an average rather than the upper bound of population FPR when we choose the threshold this way.

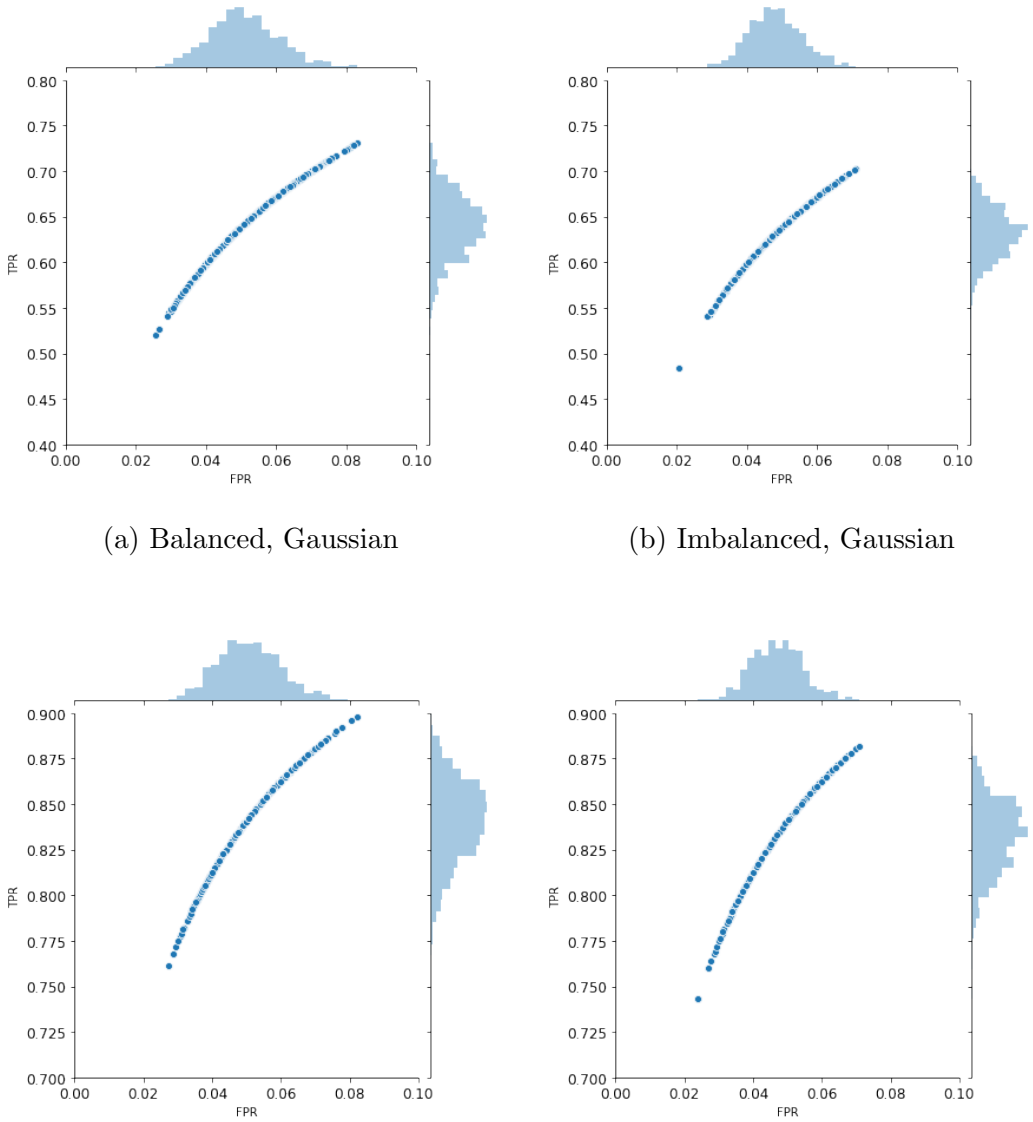
	FPR	TPR	Precision
Balanced, Gaussian	0.0515 ± 0.0100	0.6407 ± 0.0360	0.9263 ± 0.0096
Imbalanced, Gaussian	0.0488 ± 0.0073	0.6327 ± 0.0273	0.5921 ± 0.0259
Balanced, Beta	0.0516 ± 0.0100	0.8419 ± 0.0241	0.9425 ± 0.0090
Imbalanced, Gaussian	0.0469 ± 0.0076	0.8311 ± 0.0205	0.6649 ± 0.0305

Table 3.1: Population FPR and TPR mean and standard deviation when using threshold corresponding to empirical FPR= 0.05

3.3 NP classification for more control over population false positive rate (FPR)

In the last section, we understand that the classical approach of choosing a threshold for a model based on empirical measures can be misleading, and population TPR, FPR, and precision can differ from our expectations. In this section, we continue to explore the simulation data sets with the NP classification algorithm to see how things can be improved.

We construct $1,000$ NP classifiers using the identity function base classifier with $M = 11$ and $\alpha = 0.05$. Since the scoring classification function we use is the same identity function for all training data, the majority vote step in the NP umbrella algorithm can be translated to a single final threshold that is the median of the $M = 11$ thresholds for each individual NP classifier, where M is the number of NP classifiers in the NP ensemble algorithm and α is the predetermined type I error upper bound. In Figure 3.6, we plot the distribution of population FPR and TPR using this final threshold from NP algorithm. Compared to Figure 3.5, it is obvious that the distribution of population FPR is shifted to the left, with most NP classifiers having population FPR below the desired level of 0.05. With the NP classification method, we can have more control over the upper bound of FPR and be assured that the population FPR would not exceed a predetermined level with high probability.



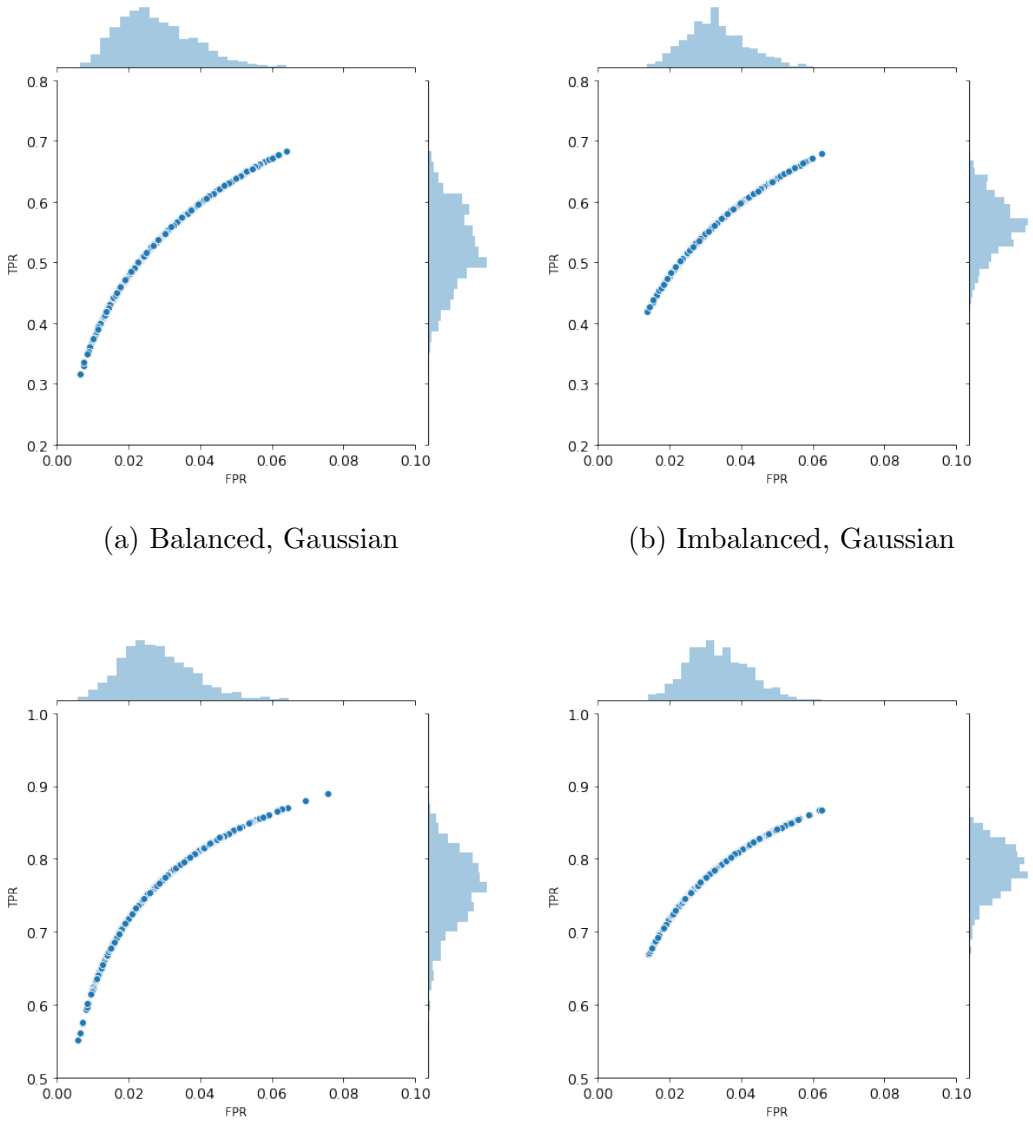
(a) Balanced, Gaussian

(b) Imbalanced, Gaussian

(c) Balanced, Beta

(d) Imbalanced, Beta

Figure 3.5: Distribution of population FPR and TPR using threshold corresponding to empirical FPR = 0.05



(a) Balanced, Gaussian

(b) Imbalanced, Gaussian

(c) Balanced, Beta

(d) Imbalanced, Beta

Figure 3.6: Distribution of population FPR and TPR using threshold from NP algorithm setting high-probability upper bound of FPR to 0.05

It is evident from Figure 3.5 and 3.6 that one of the most significant advantages of using an NP classifier is that we can have more control over FPR upper bound. In addition, if what we care more is high TPR, we can simply swap the definition of positive and negative so that the NP algorithm will put a high-probability upper bound on FNR.

3.4 NP-ROC

Comparing Figure 3.5 and 3.6, we see that as the distribution of population FPR is shifted to the left, the distribution of population TPR is inevitably shifted downwards (along the oracle ROC curve). As much as we would like to put an upper bound on the FPR, we do not want to jeopardize TPR to an undesirable level. NP-ROC curves efficiently summarize the high-probability lower and upper bounds of conditional TPR at each level of α (high-probability upper bound of FPR), so that we can make more informed decision making on the model and threshold selection (i.e. what α level would be best for our business need). For example, if we want lower bound of conditional TPR to be higher than 0.75, we should choose FPR upper bound to be 0.1.

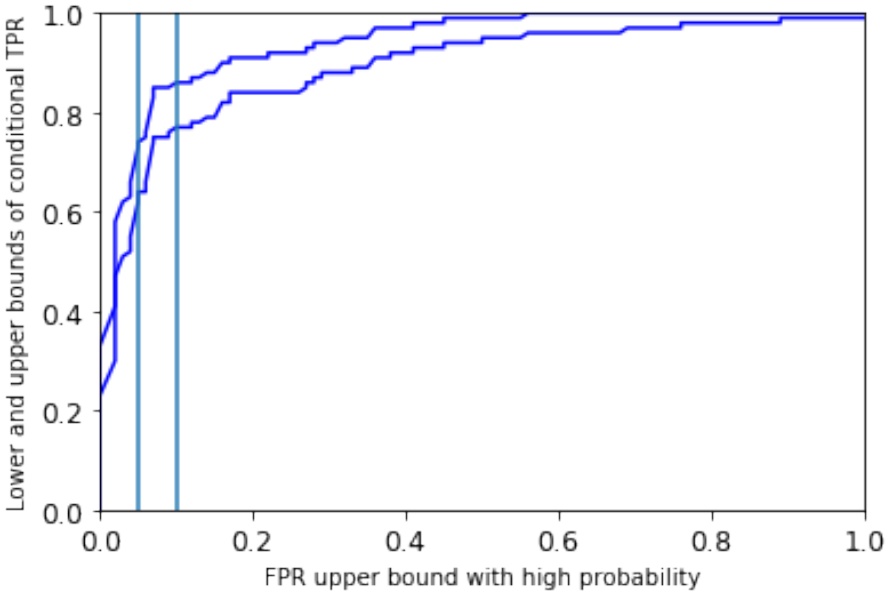


Figure 3.7: NP-ROC plot for NP classifier with identity function as base classifier for balanced Gaussian data set. Two vertical lines at $x = 0.05$ and $x = 0.1$

CHAPTER 4

Credit card fraud detection

4.1 Data

The data set comes from a Kaggle dataset, which is collected and analyzed by Dal Pozzolo et al. (2014). It contains credit card transactions by European cardholders in two days during September 2013. There are in total 30 predictors - relative time, amount of transaction, as well as 28 numerical predictors coming from principal component analysis (PCA) transformation due to confidentiality concerns. Out of the 284,807 data points, 492 ($\sim 0.17\%$) are fraudulent (class 1). This is a scenario where the data set is highly skewed, with the rare class (fraudulent class) being the more important class for the prediction.

4.1.1 Classical classification methods

We apply penalized logistic regression (penLR) with L2 regularization, random forest (RF) and support vector machine (SVM) classifier to the data. Since the negative to positive class ratio is approximately 580 : 1, we choose to use a combination of random undersampling and SMOTE oversampling to obtain a new balanced training set. We note that this is not the only way to obtain a balanced data set. We can use only undersampling (NearMiss, random undersampling, or other undersampling methods), only oversampling (SMOTE, random oversampling, or other oversampling methods), or a combination of both. Since a comparison between the performances of different sampling methods is beyond the scope of this paper, we only present the random undersampling plus SMOTE oversampling method here for the purpose of creating a balanced dataset. 3-fold cross-validations are performed on the training set for each algorithm for optimal hyperparameter search. The classifiers with

the best hyperparameters are then fitted with the whole training set to obtain a model for each method. The models are evaluated empirically using a hold-out (skewed) test set that accounts for 20% of the original data set. From the empirical ROC and PR curves (Figure 4.1 and 4.2), random forest appears to be the best classifier among the classical classifiers.

From Figure 4.3, we can have a more intuitive understanding of why it is the case. Comparing Figure 4.3(b) with 4.3(a), we see that random forest algorithm does a better job assigning lower scores to negative instances (with very few negative instances having score beyond 0.8). In the context of imbalanced classification, the large size of the negative class means that even a small proportion of negative instances having high scores would lead to a significant drop in precision.

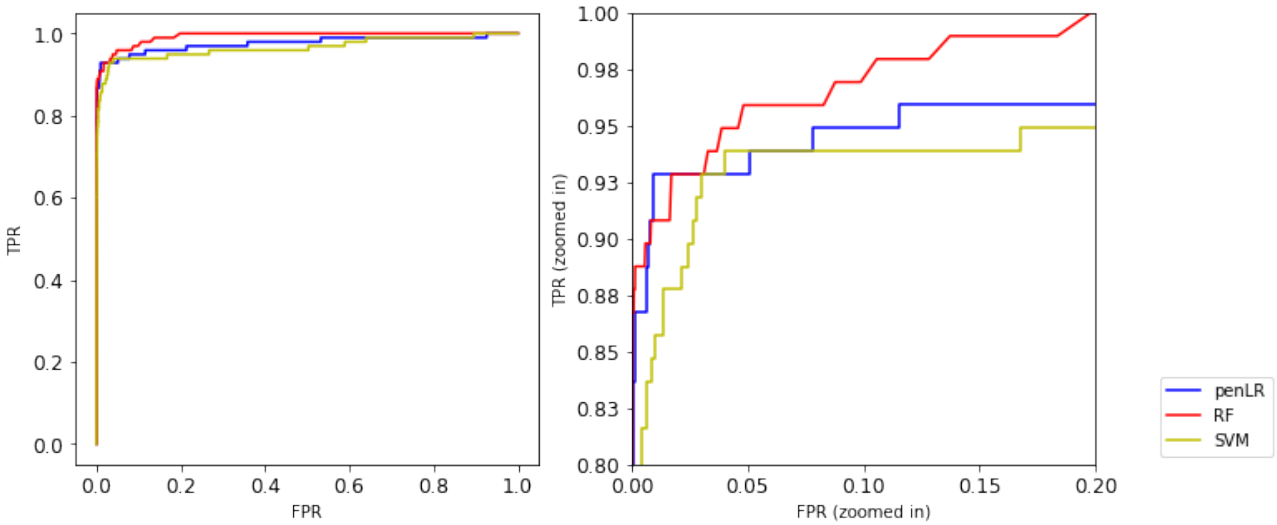


Figure 4.1: Empirical ROC curves for the classical classifiers

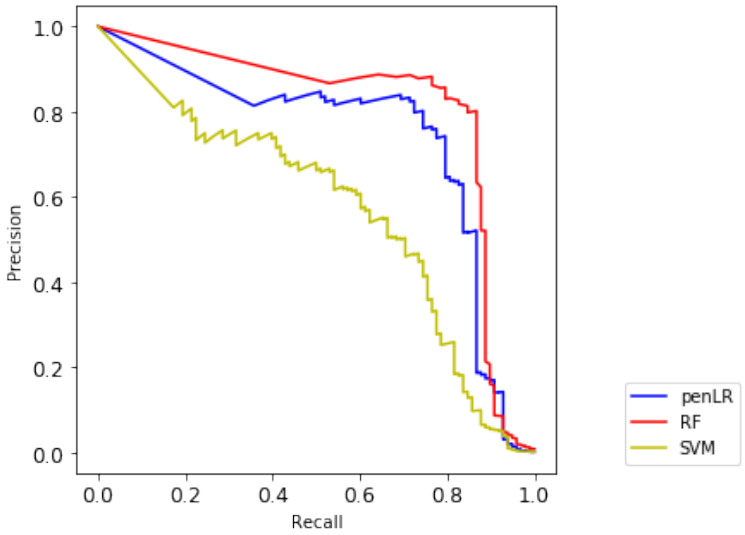
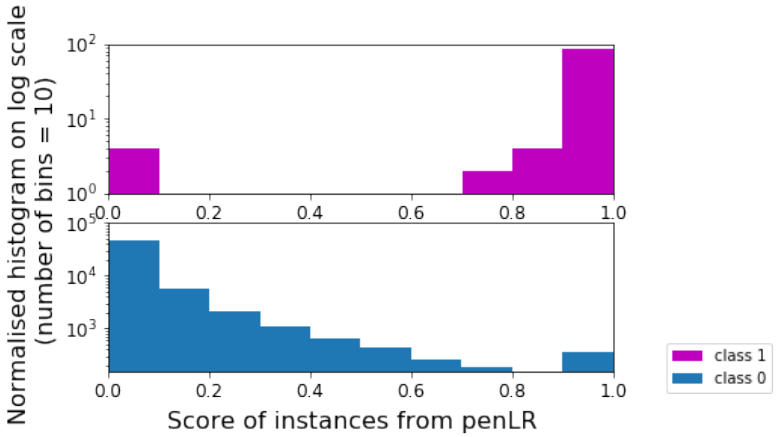
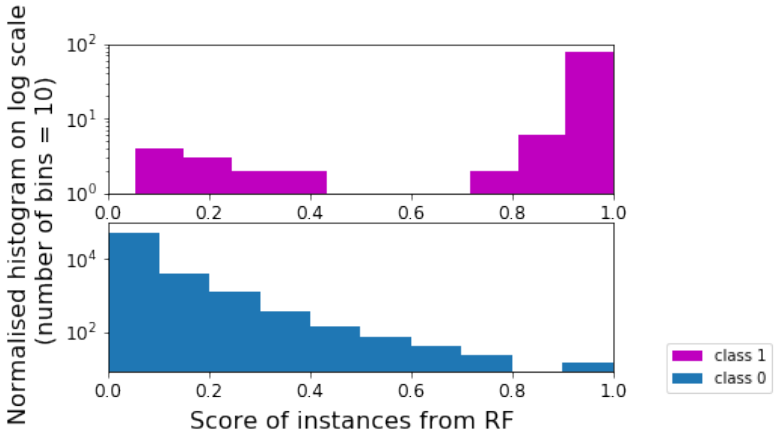


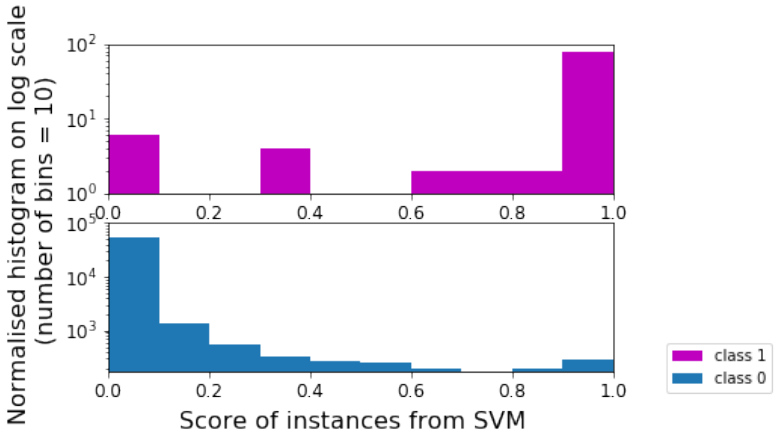
Figure 4.2: Empirical PR curves for the classical classifiers



(a)



(b)



(c)

Figure 4.3: Histogram of scores for test instances from different classifiers

4.2 Neyman-Pearson classification methods

We use the classical classifiers from the last section as the base classifiers to construct the NP umbrella algorithm as described in Section 2.1.2. We try to fit the base classifiers with (1) balanced training set as in the last section, and (2) imbalanced training set which only has its majority class under-sampled for computational feasibility.

4.2.1 Comparison with classical methods based on empirical summary statistics

We first compare the NP classifiers with the classical classifiers that are built with target FPR of 0.05 based on their empirical summary statistics evaluated on the hold-out test data set (Table 4.1). In Table 4.1, the class balance indicates the balance of the data that is used for the training of the scoring-type base classifier. The initial skewed data is balanced after SMOTE oversampling. For NP classifiers, the target FPR of 0.05 simply means the input parameter α is set to 0.05. For classical classifiers, thresholds are chosen using the empirical ROC curve at the level where empirical FPR on the training set is 0.05. Table 4.1 shows NP algorithms do push FPR to lower levels, and that although NP algorithms can deal with skewed data sets, the performance is generally further improved when NP algorithms are trained on balanced data sets.

Comparing classical and NP algorithms with balanced input in Table 4.1, we see that for RF and LR, NP algorithm can lower the empirical FPR without lowering empirical TPR. However, this phenomenon is not guaranteed. For SVM, NP algorithm lowers both empirical FPR and TPR. In general, in order to have stricter control over the FPR upper bound, NP algorithm tends to be a more "pessimistic" classifier which lowers FPR at the cost of possibly lowering TPR. This can also be seen from the difference between Figure 3.5 and Figure 3.6, with the distributions of TPR and FPR both shifted towards smaller values in Figure 3.6.

Classical/NP	**Balance**	**Classification method**	**TPR**	**FPR**	**PPV**	$\boldsymbol{F_1}$
Classical	Balanced	SMOTE + RF	0.949	0.046	0.034	0.066
Classical	Balanced	SMOTE + LR	0.929	0.046	0.033	0.064
Classical	Balanced	SMOTE + SVM	0.939	0.048	0.033	0.063
NP	Skewed	NP + RF	0.918	0.034	0.044	0.084
NP	Skewed	NP + LR	0.929	0.038	0.040	0.077
NP	Skewed	NP + SVM	0.908	0.038	0.039	0.075
NP	Balanced	NP + SMOTE + RF	0.949	0.038	0.041	0.079
NP	Balanced	NP + SMOTE + LR	0.929	0.040	0.039	0.075
NP	Balanced	NP + SMOTE + SVM	0.918	0.035	0.043	0.081

Table 4.1: Empirical summary statistics evaluated on a hold-out test set. Threshold chosen (classifier constructed) with a target FPR=0.05.

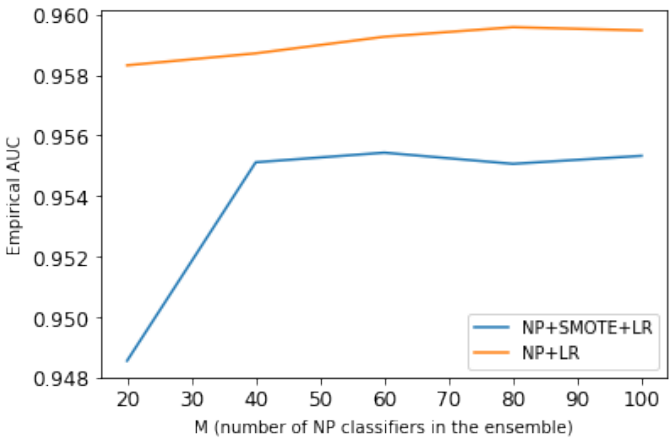
In addition, using the mean vote of individual NP classifiers as a score, we can make NP classifiers scoring type classifiers as well. Here we measure the empirical AUC of the NP classifiers with $M = 100$. NP classifiers do not show superiority over the classical classifiers from the perspective of empirical AUC (Table 4.2).

Classical/NP	**Balance**	**Classification method**	**AUC**
Classical	Balanced	SMOTE + RF	0.993
Classical	Balanced	SMOTE + LR	0.976
Classical	Balanced	SMOTE + SVM	0.955
NP	Skewed	NP + RF	0.970
NP	Skewed	NP + LR	0.960
NP	Skewed	NP + SVM	0.979
NP	Balanced	NP + SMOTE + RF	0.980
NP	Balanced	NP + SMOTE + LR	0.955
NP	Balanced	NP + SMOTE + SVM	0.957

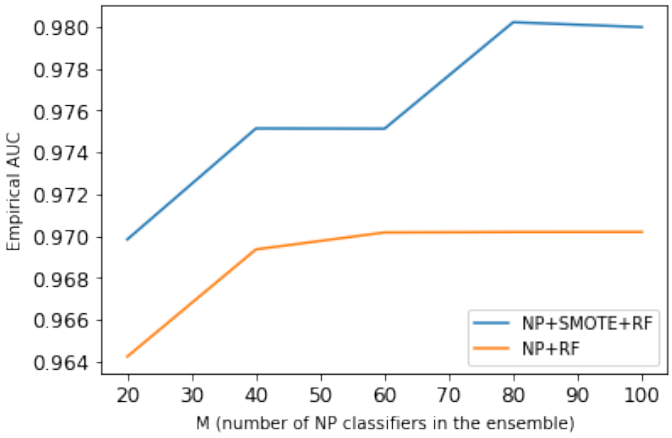
Table 4.2: Empirical AUC evaluated on a hold-out test set. Threshold chosen (classifier constructed) with a target FPR=0.05.

4.2.2 NP Classifiers as scoring-type classifiers

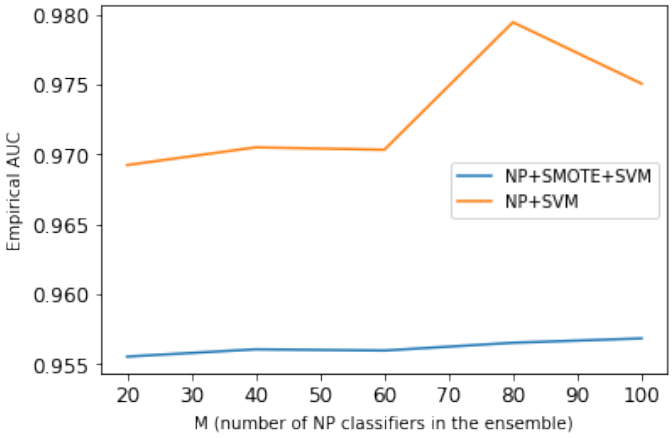
NP classifiers can be seen as scoring-type classifiers by taking the mean vote of M individual NP classifiers as the score. We increase the value of M and observe an increase in empirical AUC evaluated on the hold-out test set (Figure 4.4).



(a)



(b)



(c)

Figure 4.4: Empirical AUC increases with M for NP classifiers

4.2.3 Comparing performance of NP Classifiers using NP-ROC

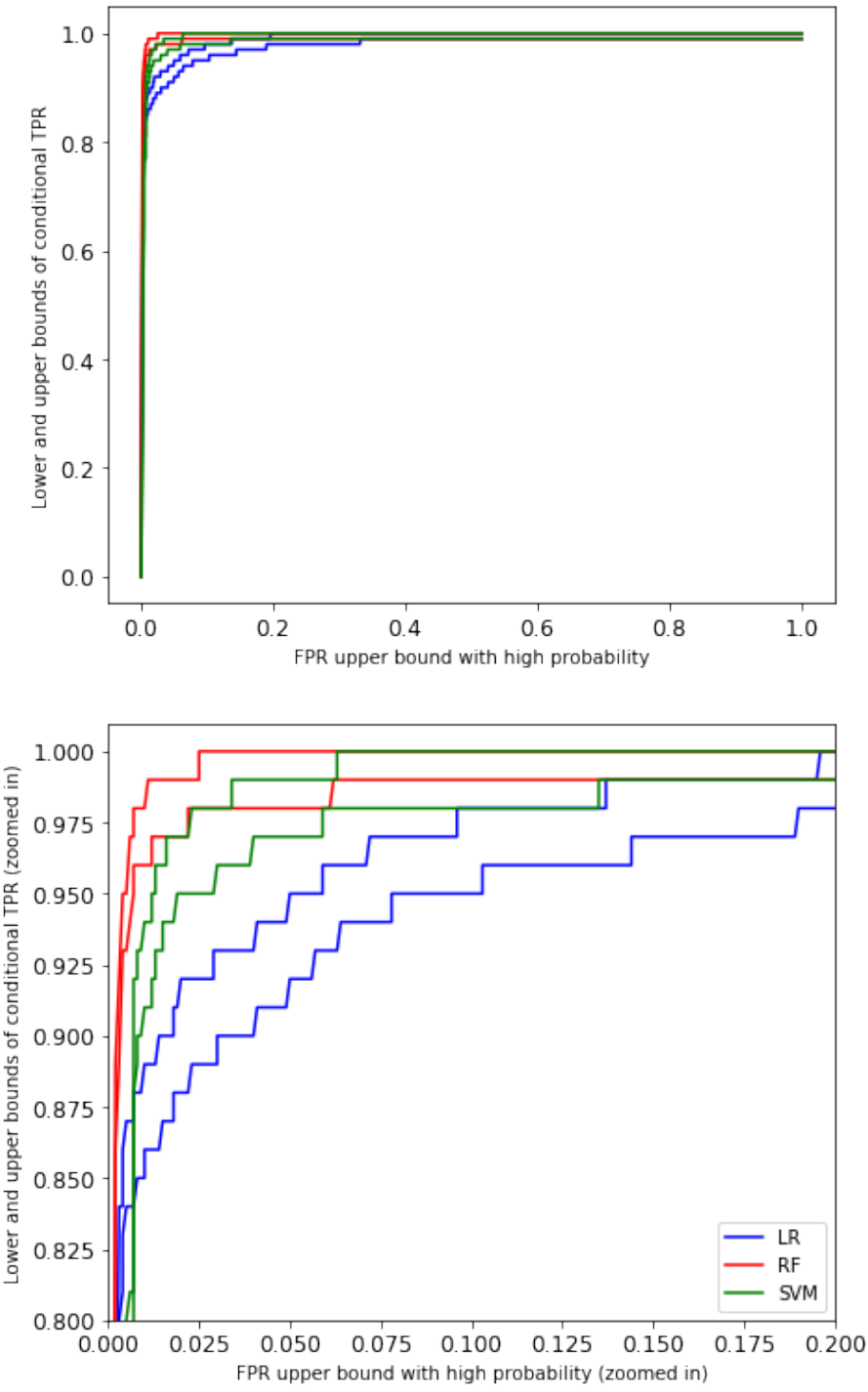


Figure 4.5: NP-ROC intervals for NP classifiers with balanced training sets

The NP-ROC curves for the three NP classifiers constructed with balanced training sets are shown in Figure 4.5. NP classifier with RF as the base scoring function has the best performance.

4.2.4 Making decisions on FPR upper bound with NP-ROC

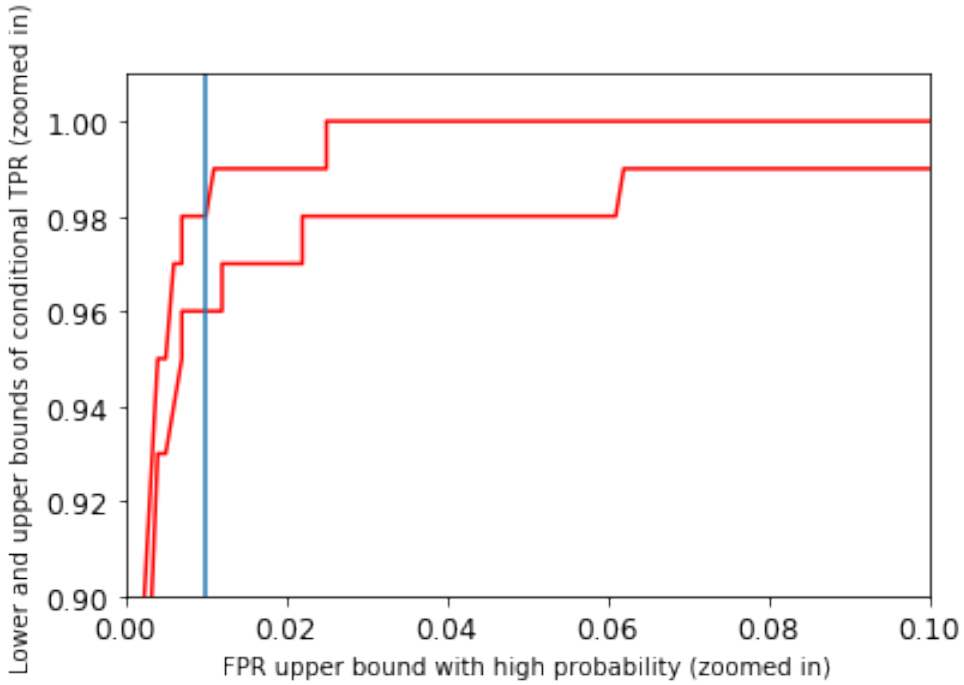


Figure 4.6: NP-ROC for NP classifiers with RF scoring function. Vertical line at $x = 0.01$.

From the NP-ROC plot (Figure 4.6), we note that for upper bound FPR between 0.03 and 0.05, the lower and upper bounds of conditional TPR is the same. Thus to choose FPR= 0.03 over FPR= 0.05 is preferable. If we further limit the FPR upper bound to 0.01, we still get a relatively high conditional TPR with lower bound 0.96 and upper bound 0.98.

CHAPTER 5

Conclusion

In this paper, we review the classical and NP classification algorithms and evaluation measures. We use simulation data sets to illustrate the discrepancy between empirical measures evaluated on the test set and actual measures evaluated on the whole population. We implement the NP classification algorithm and show how it has more effective control type I error under a desirable level compared to classical classification algorithms.

We apply both classical and NP classification to the credit fraud data set and compare their performance. From the experiments and data analysis, we have the following conclusions.

- Random forest classifier seems to perform the best both as a classical classifier and as a base classifier for NP algorithm.
- NP classifiers can work with the imbalanced training set, although re-sampling the data set to be balanced might improve the performance further for some base classifier.
- Treating mean vote of individual NP classifiers as score, we can see NP classifiers as scoring-type classifiers as well. The empirical AUC of such classifiers tends to increase with the number of individual NP classifiers in the ensemble classifier. However, the empirical AUC of such classifiers does not outperform that of the classical classifiers.

In summary, NP classifiers control the FPR upper bound effectively, which can mean that both FPR and TPR are pushed to a lower level. This method is most applicable to scenarios where we need a really strict upper bound on FPR. The FPR upper bound should be determined by closely examining the NP-ROC plot, as when two FPR upper bounds have

similar lower and upper bounds of conditional TPR, the smaller FPR upper bound is always preferable.

Bibliography

A. Amin, S. Anwar, A. Adnan, M. Nawaz, N. Howard, J. Qadir, A. Hawalah, and A. Hussain. Comparing oversampling techniques to handle the class imbalance problem: A customer churn prediction case study. *IEEE Access*, 4:7940–7957, 2016. ISSN 2169-3536. doi: 10.1109/ACCESS.2016.2619719.

K. W. Bowyer, N. V. Chawla, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal Of Artificial Intelligence Research*, Volume 16, pages 321-357, 2002.

A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Systems with Applications*, 41:4915–4928, 08 2014. doi: 10.1016/j.eswa.2014.02.026.

H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, Sep. 2009. ISSN 2326-3865. doi: 10.1109/TKDE.2008.239.

N. Japkowicz and M. Shah. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, 2011. doi: 10.1017/CBO9780511921803.

T. Saito and M. Rehmsmeier. The Precision-Recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3):1–21, 03 2015. doi: 10.1371/journal.pone.0118432. URL `https://doi.org/10.1371/journal.pone.0118432`.

X. Tong, Y. Feng, and J. J. Li. Neyman-Pearson classification algorithms and NP receiver operating characteristics. *Science Advances*, 4(2), 2018. doi: 10.1126/sciadv.aao1659.

J. Zhang and I. Mani. KNN approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Datasets*, 2003.