A Novel Penalized Log-likelihood Function for Class Imbalance Problem

Lili Zhang
Analytics and Data Science Ph.D. Program

Follow this and additional works at: https://digitalcommons.kennesaw.edu/dataphd_etd

Part of the Business Analytics Commons, and the Statistics and Probability Commons

Recommended Citation
https://digitalcommons.kennesaw.edu/dataphd_etd/5
Dissertation Defense Outcome

Name  lili zhang  KSU ID  000783396
Email  lzhang18@students.kennesaw.edu  Phone Number  (470) 865-2083
Program  Analytics and Data Science Ph.D. Program

Title
A Novel Penalized Log-likelihood Function for Class Imbalance Problem

Dissertation Defense:  Date  03/24/2020
✓ Passed  □ Failed  □ Passed With Revisions (attach revisions)

Signatures

Herman Ray  March 26, 2020  Date
Committee Chair

Joe DeMaro  Date
Date

Committee Member

Sherry Xi  Date
Date

Committee Member

Ying Xie  Date
Date

Committee Member

Dr Sherrill Hayes  Date
Date

Program Director

Jennifer Priestley  Date
Date

Department Chair

Graduate College

Graduate College Approval

Updated 9/11/19
A Novel Penalized Log-likelihood Function for Class Imbalance Problem

A Dissertation Presented for the
Doctor of Philosophy
Degree
Kennesaw State University

Lili Zhang
May 2020
Acknowledgments

It has been a blessing to finish my Ph.D. study and research in the Analytics and Data Science Ph.D. Program at Kennesaw State University (KSU). My sincere thanks go to every professor, coworker, and classmate I have worked with. Their support has greatly boosted my performance.

My advisor, Dr. Herman Ray, instructed me to think thoroughly and ahead at each step of my dissertation research from the method development, to the experimental analysis, to the manuscript preparation. He always provided wise advice on how to properly handle academic issues, for example, how to carefully and respectfully address the comments from the reviewers. Our executive director, Dr. Jennifer Priestley, recruited me into the Ph.D. program four years ago. When I arrived in the United States without a place to stay, she kindly drove me around and helped find a brand-new apartment where I stayed throughout my Ph.D. program. She was the instructor of my favorite binary classification course and was also the co-author of my three published articles. No matter how busy her schedule was, she always read my manuscripts in detail and even pointed out some grammatical errors. Dr. Ying Xie advised me in my first year on the text mining project of discovering sentiment-change-driven events based on social media data, which was my first published article and gave me the confidence to become a better researcher. Dr. Joseph DeMaio was the advising faculty whom I worked closely with on the project of reducing travel times for the Cobb County Fire Department. Through that project, I learned how to be a socially responsible researcher. Dr. Sherry Ni was the instructor of data mining courses, and Dr. Mingon Kang was the instructor of big data analytics and machine learning courses. They patiently addressed every question I asked in class. I was probably the student who asked the hardest questions in their classes. Dr. Lin Li joined my committee at the last minute after Dr. Kang
left KSU for a new faculty position. She spent the whole morning listening to my dissertation research proposal presentation and discussing it with me. Dr. Sherrill Hayes, our program director, always helped monitor my progress and made sure that I was making good progress toward graduation. Dr. Lewis VanBrackle, the Chair of the Department of Statistics and Analytical Sciences, trusted me and provided me the opportunity to teach a Graduate R Programming course as an instructor. Ms. Cara Reeve, our program administrator, helped handle all the documents and guided me through the logistics of getting my degree. Dr. Chih-Cheng Hung and Dr. Xiaohua Xu from the Department of Computer Science helped read my manuscript and provided suggestions for publication. Mr. David Schmidt, Director of the ESL Center, helped proofread my dissertation draft word by word. The support from college and university leaders, Dr. Pamela Whitten, Dr. Mike Dishman, Dr. Mark Forehand, Dr. David Baugher, Dr. Nikki Palamiotis, Dr. Phaedra Corso, Dr. Jon Preston, Dr. Marla Bell, and all, enabled me to be successful in my Ph.D. program.

I also appreciate the collaborations and friendship with all of the Ph.D. students in my program. Mr. Trent Geisler, a new Ph.D. student in our program, kindly shared his thoughts and experiences on the application of my research in the marketing area, in which he used to work. That broadened my research insights and helped my conversations for getting the marketing data science position at HPE after graduation. It was very nice to work closely on the coursework and projects with my cohort, Mr. Shashank Hebbar, Ms. Jessica Rudd, Ms. Yan Wang, Mr. Yiyun Zhou, as well as the first cohort, Dr. Edwin Baidoo, Mr. Sergiu Buciumas, Dr. Bogdan Gadidov, Dr. Jie Hao, Dr. Linh Le, and Dr. Bob Vanderheyden. Mr. Sanjoosh Akkineni and Ms. Seema Sangari were my great team members in the MagMutual research lab. Besides the study and research, it was also a source of joy and achievements for me to serve as the secretary of the Analytics and Data Science Organization and work with Mr. Sanjoosh Akkineni, Ms. Jessica Rudd, Mr. Srivatsa Mallapragada, and Ms. Srivarna Settisara Janney.

Besides people in the KSU community, I would like to thank Mr. Soon Tan, CEO of Ermas Consulting Inc., for generously sponsoring the Ermas research lab I worked in for two academic years from 2017 to 2019 with the accumulated funding of $134,000. I gained plenty of industry experiences by working with him, his team, and his partners,
which was extremely helpful for my future career. I also would like to present my thanks to all conferences and publication venues that accepted my work: Journal of Applied Statistics, International Journal of Database Management Systems, INFORMS Annual Meeting 2019, INFORMS Business Analytics Conference 2019, Southern Data Science Conference, International Conference on Web Intelligence, SAS Global Forum, The Workshop of Mathematical Optimization of Systems Impacted by Rare and High-Impact Random Events, YinzOR Student Conference, The Workshop of Emerging Data Science Methods for Complex Biomedical and Cyber Data, The Workshop of Women in Data Science, KSU Analytics Day, KSU R Day, and HPCC Systems Summit Community Day. It has been my honor to present my work and communicate with other participants. My appreciation also goes to the researchers who kindly spent their time discussing the research work with me even after the conferences we attended: Dr. Xiaoguang Tian from Purdue University Fort Wayne and Dr. Hongxia Yin from Minnesota State University Mankato.

Family and friends are the most precious part of my life. Without their understanding and support, I would not be able to make it. My parents, Mr. Zhonghua Zhang and Ms. Qingping Wang, did their best to raise me and educate me although they left this world when I was young. My mom always told me to be a useful person for the society. At this moment, I would like to tell her, “Mom, I followed your words and did it”. I will keep doing it in the future and teach my baby to do it – be a useful person for the society. My dear baby, Allen Tong, arrived in my life two years ago. I would always remember the moments that I worked next to his crib when he slept quietly. My husband, Dr. Yang Tong, supported every decision I made for pursuing my study and career in the past decade. My grandparents, uncles, aunts, and cousins took care of me after my mom passed away when I was thirteen. I owe them a big thanks. My cousin, Dr. Hao Li, taught me to be a professionally creditable and responsible person and build a good reputation for the career development. Dr. Xiaohong Wang was my best friend in the past decade. She started a Ph.D. program in Singapore at the same time as I, and completed two months earlier. Dr. Yilu Liu and her parents always kindly hosted me when I visited them in Tennessee and asked about me when I studied at KSU.
Abstract

The log-likelihood function is the optimization objective in the maximum likelihood method for estimating models (e.g., logistic regression, neural network). However, its formulation is based on assumptions that the target classes are equally distributed and the overall accuracy is maximized, which do not apply to class imbalance problems (e.g., fraud detection, rare disease diagnoses, customer conversion prediction, cybersecurity, predictive maintenance). When trained on imbalanced data, the resulting models tend to be biased towards the majority class (i.e. non-event), which can bring great loss in practice. One strategy for mitigating such bias is to penalize the misclassification costs of observations differently in the log-likelihood objective function in the learning process. Existing penalized log-likelihood functions require either hard hyperparameter estimation or high computational complexity. In the present work, we propose a novel penalized log-likelihood function by including penalty weights as decision variables for observations in the minority class (i.e. event) and learning them from data along with model coefficients/parameters. The proposed log-likelihood function is applied to train logistic regression and neural network models, which are compared with models trained by existing penalized log-likelihood functions on 10 public imbalanced datasets. The model performance is measured by the statistics of Area under ROC Curve (i.e. AUROC or AUC) over repeated runs of 10-fold stratified cross validation, including 95% confidence interval, mean and standard deviation, as well as the training time. A more detailed analysis is conducted to examine the estimated probability distributions and additional performance measurements (i.e. Type I error, Type II error, accuracy) under the chosen probability cutoff. The results demonstrate that the discrimination ability of the models is improved by using the proposed log-likelihood function as the learning objective while reducing or maintaining the computational complexity compared with existing ones.
# Table of Contents

1 Introduction ............................................ 1  
   1.1 Motivation ........................................ 1  
   1.2 Contribution ....................................... 3  

2 Literature Review ........................................ 4  
   2.1 Class Imbalance Problem ............................... 4  
   2.1.1 Problem Definition ................................ 4  
   2.1.2 Challenging Issue ................................ 5  
   2.1.3 Solution Methods ................................ 7  
   2.1.4 Performance Measurements ......................... 10  
   2.2 Maximum Likelihood Estimation ......................... 23  
   2.2.1 Logistic Regression Model Estimation ............... 23  
   2.2.2 Penalized Log-likelihood Objective Functions for Logistic Regression . 28  
   2.2.3 Neural Network Model Estimation .................. 31  
   2.2.4 Penalized Log-likelihood Objective Functions for Neural Network .. 36  
   2.3 Model Evaluation .................................... 37  
   2.3.1 Resampling Strategies ............................. 37  
   2.3.2 Machine Learning Bias and Variance ................ 39  
   2.3.3 Statistical Bias and Variance ....................... 40  

3 A Novel Penalized Log-likelihood Objective Function .......... 45  
   3.1 Definition for Logistic Regression ..................... 45  
   3.2 Learning by Gradient Descent ......................... 46
Appendix

A Mathematical Preliminaries ........................................ 103
A.1 Empirical Logit Plotting ........................................... 103
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Performance of Models under Event Rate 0.12%</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Performance of Models under Event Rate 50%</td>
<td>6</td>
</tr>
<tr>
<td>2.3</td>
<td>An Example of Test Data</td>
<td>11</td>
</tr>
<tr>
<td>2.4</td>
<td>Confusion Matrix</td>
<td>11</td>
</tr>
<tr>
<td>2.5</td>
<td>Predicted Classes under Probability Cutoff 0.7</td>
<td>14</td>
</tr>
<tr>
<td>2.6</td>
<td>Confusion Matrix of the Example Test Data</td>
<td>14</td>
</tr>
<tr>
<td>2.7</td>
<td>Single-Value Performance Metrics</td>
<td>15</td>
</tr>
<tr>
<td>2.8</td>
<td>Constructing ROC Curve for the Example Test Data</td>
<td>16</td>
</tr>
<tr>
<td>2.9</td>
<td>Observations in Descending Order of Predicted Probability for Positive Class</td>
<td>18</td>
</tr>
<tr>
<td>2.10</td>
<td>Constructing KS Chart for the Example Test Data</td>
<td>18</td>
</tr>
<tr>
<td>2.11</td>
<td>Constructing Gain and Lift Charts for the Example Test Data</td>
<td>20</td>
</tr>
<tr>
<td>2.12</td>
<td>Cost Matrix</td>
<td>37</td>
</tr>
<tr>
<td>2.13</td>
<td>Bias-Variance Decompositions</td>
<td>41</td>
</tr>
<tr>
<td>3.1</td>
<td>Comparison of Penalized Log-likelihood Functions</td>
<td>49</td>
</tr>
<tr>
<td>4.1</td>
<td>Basic Characteristics of Datasets</td>
<td>55</td>
</tr>
<tr>
<td>4.2</td>
<td>Experimental Results of Logistic Regression Models</td>
<td>58</td>
</tr>
<tr>
<td>4.3</td>
<td>Variables for Analysis and Modeling</td>
<td>62</td>
</tr>
<tr>
<td>4.4</td>
<td>Frequency of Dependent Variable</td>
<td>63</td>
</tr>
<tr>
<td>4.5</td>
<td>Information Values</td>
<td>64</td>
</tr>
<tr>
<td>4.6</td>
<td>Performance Measurements on Validation Data and Test Data</td>
<td>66</td>
</tr>
<tr>
<td>4.7</td>
<td>Estimated Model Coefficients</td>
<td>67</td>
</tr>
<tr>
<td>4.8</td>
<td>VIF for NumberOfTime60 − 89DaysPastDueNotWorse</td>
<td>68</td>
</tr>
</tbody>
</table>
4.9 The Estimated Probability Bins with the Most Observations. . . . . . . . . 70
4.10 Bias-Variance Decomposition of Squared Loss . . . . . . . . . . . . . . . . 73
4.11 Bias-Variance Decomposition of 0-1 Loss . . . . . . . . . . . . . . . . . . 74
5.1 Experimental Results of Neural Network Models. . . . . . . . . . . . . . . 78
5.2 Frequency of Dependent Variable in the Dataset Ecoli. . . . . . . . . . . . 80
5.3 Performance Measurements on Validation Data and Test Data. . . . . . 82
List of Figures

2.1 Sampling ................................................................. 7
2.2 Cost-Sensitive Learning .................................................. 9
2.3 ROC Curve of the Example Test Data ................................. 16
2.4 KS Chart of the Example Test Data ................................. 19
2.5 Gain Chart of the Example Test Data ................................. 20
2.6 Lift Chart of the Example Test Data ................................. 21
2.7 Sensitivity, Specificity, and Accuracy vs. Probability Cutoff ........ 22
2.8 Youden Index vs. Probability Cutoff ................................. 23
2.9 Logistic Regression ........................................................ 24
2.10 Sigmoid Function ......................................................... 25
2.11 Neural Network ............................................................ 32
2.12 10-fold Cross Validation ................................................. 38
2.13 Bootstrapping ............................................................. 38

4.1 Searching Optimal $\tau$ ................................................... 57
4.2 Empirical Logit Plots of the Dataset Wine_Quality .................... 60
4.3 Predicted Probabilities and Probability Cutoff on Test Data ........... 65
4.4 Empirical Logit Plot of NumberOfTime60 – 89DaysPastDueNotWorse 68

5.1 ROC Curves of standard vs. learnable .................................. 79
Chapter 1

Introduction

1.1 Motivation

The research work was motivated by challenges faced in the anti-money laundering (i.e. AML) and bankruptcy prediction problems from industry clients in 2017.

In summer 2017, as an intern on a team with members from Ermas Consulting Inc., Mizen Group, and DXC Technology on developing AML models for a leading insurance company, the conversations with experienced AML experts revealed to me the current challenges in this field – an extremely large number of false positive cases and very high labor costs. A part of the reason was that the analysts relied on rule-based models and scenario analysis to detect AML activities for the regulation purpose. As the concept of data science became more popular, it was hoped to incorporate data science techniques to reduce false positive rates while maintaining or improving true positive rates. As known, sufficient historical data was expected to train a good predictive model. However, in the AML data, there were very few observations flagged with the money laundering in the database, although there were so many unflagged observations, either truly normal or undetected. This problem was referred to as the class imbalance problem in the literature, which was a category of problems of classifying the imbalanced data with an unequal target class distribution. By definition, the imbalanced data was the dataset where observations in the majority class (i.e. event) were many more than the minority class (i.e. non-event). In the context of the anti-money problem, the majority class was normal and the minority class was money laundering.
In the meantime, as a Ph.D. student in Analytics and Data Science at Kennesaw State University, the bankruptcy prediction project in Dr. Priestley’s binary classification class and Dr. Ni’s data mining class in 2017 presented the same challenge of class imbalance problems. The dataset was provided by a national credit bureau. The proportion of bankruptcy observations was 0.12% while the proportion of non-bankruptcy was 99.88%. The standard statistics and machine learning models did not perform well. Although the overall accuracy was as high as 99%, almost all bankruptcy observations were misclassified as non-bankruptcy, which would bring great loss in practice. To understand the nature of the imbalanced data and the reason the standard models did not work well, an experimental study was conducted to show how the different event rates (i.e. the proportion of the minority class) influenced discrimination abilities of seven bankruptcy prediction models [115]. The results of this study demonstrated two assumptions of the training objective function formulations for standard statistics and machine learning models: 1) the target classes were equally distributed; 2) the misclassifications of target classes are equal. Under these assumptions, the objective essentially maximized the overall accuracy.

Class imbalance problems existed in almost every area (e.g., credit scoring, customer churn prediction, defective product detection, rare disease diagnosis, cybersecurity, fraud detection). Because of the significance of the problem, it would be meaningful to develop a novel solution method. In the related literature, an existing solution method was the cost-sensitive learning, which was about adjusting the optimization objective (e.g., log-likelihood function) by assigning different penalty weights to misclassification costs of observations in the training process. That was very interesting for me, because the concentration of my Master’s study was the optimization. King and Zeng proposed a penalized log-likelihood objective function to train logistic regression in rare event data or imbalanced data [57]. In this penalized log-likelihood function, class penalty weights were determined by the estimated population proportion of the minority class and its sample proportion. As known, it was very hard to accurately estimate the population proportion [23], which ultimately influenced the model performance, as demonstrated in a related empirical study showing how the different class penalty weights influenced the performance of logistic regression models on the imbalanced data [116]. The next question became how to intelligently determine the penalty
weights. For that, Deng used the Gaussian kernel [21], as well as Maalouf and Trafalis [73].

But that presented the challenge of high computational complexity $O(n^3)$, which did not meet the requirements of big data. Inspired by a discussion with Dr. Xie on the choice of hyperparameters used in a deep learning model we were working on, Dr. Ray and I had a conversation about learning a parameter from data. We started to think about treating class penalty weights as decision variables in the log-likelihood function and learning them from data along with model coefficients/parameters, instead of pre-defining them and plugging them in the log-likelihood function as constant values. Experiments on this idea showed promising results, which ended up to be this research work.

1.2 Contribution

A novel penalized log-likelihood function is proposed to better estimate classification models for the improved differentiation ability and computation efficiency in class imbalance problems. The penalty weights for misclassification costs of observations in the minority class are included as decision variables in the log-likelihood function and learned from data along with model coefficients/parameters. This addresses the challenges of either hard hyperparameter estimation or high computational complexity faced by existing penalized log-likelihood functions for the imbalanced data, where penalty weights are pre-defined by some rules and plugged in the log-likelihood function as constants.

The proposed approach has broad applications. It can be applied to improve the logistic regression models with the model interpretability maintained, especially for class imbalance problems in highly regulated industries (e.g., anti-money laundering, credit scoring). It can also be applied to improve the neural network models for learning more complicated relationships.
Chapter 2

Literature Review

2.1 Class Imbalance Problem

2.1.1 Problem Definition

The class imbalance problem is the problem of classifying the imbalanced data with an unequal target class distribution [3] [36]. In the imbalanced data, many more observations are labeled by the majority class (i.e. non-event) than the minority class (i.e. event). One example is the fraud detection problem with more observations in non-fraud than fraud, where the majority class is non-fraud and the minority class is fraud. Similar to generic classification problems, the class imbalance problem is usually discussed separately in two categories: 1) the binary classification with only two target classes; 2) the multi-class classification with multiple target classes. As the question “whether or not” is asked the most in the decision making process, the focus of this research is the binary classification. It can be easily extended to multi-class, since the multi-class classification can be solved by a sequence of binary ones.

Although there is no standard definition on the imbalance degree for a problem to be considered as a class imbalance problem, Weiss et. al shows that when the imbalance ratio is 2:1 or larger, the discrimination abilities of classifiers suffer from the imbalance nature of the data, based on a research study of the class imbalance impacts on the minority class classification performance on 26 binary class datasets [107]. Hence, the imbalanced
data is considered by most practitioners to be the data where the majority class has twice as many or more observations as the minority class [43]. It exists in almost every area, such as customer churn prediction [120] [4], adverse drug reaction detection [93], abnormal activity recognition [31], network intrusion detection [16], dangerous behavior recognition [7], machine fault detection [89], sentiment classification [111], software defect prediction [95], and video object detection [32].

In practice, the minority class is usually the class of interest that is more important or more costly if misclassified, such as fraud in the fraud detection problem [84], malignance in the breast cancer diagnosis problem [60], and delinquency in the credit scoring problem [13]. It is expected to differentiate the minority class effectively from the majority class by a classification model to avoid losses (e.g., money, reputation, health).

2.1.2 Challenging Issue

The standard statistics and machine learning models are biased towards the majority class and misclassify the minority class as the majority class severely, when trained on the imbalanced data [101].

To empirically show this issue and provide better understanding on it, Zhang et. al studied the influence of the event rate (i.e. the proportion of the minority class) on discrimination abilities of seven bankruptcy prediction models [115]. In the original dataset, the proportion of bankruptcy observations is 0.12%. Under this event rate 0.12%, models (e.g., decision tree, gradient boosting, random forest, neural network, support vector machine, logistic regression) misclassify more than 87% of bankruptcy observations as non-bankruptcy, as indicated by Type II Error in Table 2.1, which can bring great loss. However, as the event rate increases to 50% by resampling the data, Type II Error of all models reduce substantively, as shown in Table 2.2.
### Table 2.1: Performance of Models under Event Rate 0.12%

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>F1 Score</th>
<th>Type I Error</th>
<th>Type II Error</th>
<th>Cutoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree</td>
<td>99.88%</td>
<td>.</td>
<td>0%</td>
<td>100%</td>
<td>.</td>
</tr>
<tr>
<td>Gradient Boosting</td>
<td>99.88%</td>
<td>.</td>
<td>0%</td>
<td>100%</td>
<td>.</td>
</tr>
<tr>
<td>Bayesian Network</td>
<td>64.43%</td>
<td>0.0056</td>
<td>35.59%</td>
<td>17.31%</td>
<td>0.11</td>
</tr>
<tr>
<td>Random Forest</td>
<td>86.83%</td>
<td>0.0022</td>
<td>13.08%</td>
<td>87.95%</td>
<td>0.01</td>
</tr>
<tr>
<td>Neural Network</td>
<td>99.23%</td>
<td>0.0221</td>
<td>0.65%</td>
<td>92.94%</td>
<td>0.01</td>
</tr>
<tr>
<td>Support Vector Machine</td>
<td>99.88%</td>
<td>.</td>
<td>0%</td>
<td>100%</td>
<td>.</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>99.41%</td>
<td>0.0204</td>
<td>0.47%</td>
<td>95.01%</td>
<td>0.01</td>
</tr>
</tbody>
</table>

### Table 2.2: Performance of Models under Event Rate 50%

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>F1 Score</th>
<th>Type I Error</th>
<th>Type II Error</th>
<th>Cutoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree</td>
<td>72.26%</td>
<td>0.7507</td>
<td>38.97%</td>
<td>16.50%</td>
<td>0.28</td>
</tr>
<tr>
<td>Gradient Boosting</td>
<td>73.44%</td>
<td>0.7623</td>
<td>38.28%</td>
<td>14.84%</td>
<td>0.42</td>
</tr>
<tr>
<td>Bayesian Network</td>
<td>70.53%</td>
<td>0.7413</td>
<td>43.41%</td>
<td>15.53%</td>
<td>0.37</td>
</tr>
<tr>
<td>Random Forest</td>
<td>73.93%</td>
<td>0.7656</td>
<td>37.31%</td>
<td>14.84%</td>
<td>0.42</td>
</tr>
<tr>
<td>Neural Network</td>
<td>72.75%</td>
<td>0.7579</td>
<td>39.81%</td>
<td>14.70%</td>
<td>0.37</td>
</tr>
<tr>
<td>Support Vector Machine</td>
<td>73.23%</td>
<td>0.7605</td>
<td>38.56%</td>
<td>14.98%</td>
<td>0.49</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>72.61%</td>
<td>0.7575</td>
<td>40.36%</td>
<td>14.42%</td>
<td>0.42</td>
</tr>
</tbody>
</table>

Why is the performance of these models driven by the event rate? The reason is that the standard statistics and machine learning methods are formulated based on two assumptions to maximize the overall accuracy. First, target classes are equally distributed [51]. Second, the misclassifications of all target classes are equal [86]. However, both of these assumptions are violated on the imbalanced data. This can be reflected by results under the event rate 0.12% in Table 2.1, where the overall accuracy of most models is 86% or higher but more than 87% of bankruptcy observations are misclassified as non-bankruptcy.
2.1.3 Solution Methods

Because of significant and broad applications of class imbalance problems, researchers have made efforts to improve solutions in the past decades. Based on the workflow of the classification process, the proposed solutions are summarized in four categories, including input data, feature, algorithm, and output data.

1. Input Data: Sampling

Sampling methods modify the data to achieve a balanced target class distribution, including random oversampling [9], random undersampling [27], synthetic minority oversampling technique (SMOTE) [40] [11], and the integration of sampling and boosting [59]. The oversampling increases the number of observations in the minority class, while the undersampling reduces the number of observations in the majority class, as shown in Figure 2.1, where the circle denotes the majority class and the triangle denotes the minority class.

![Figure 2.1: Sampling](image)

2. Feature

(a) Feature Selection

Feature selection methods first rank the importance of all features (i.e. independent variables) based on statistical metrics (e.g., information gain, chi-square, and odds ratio for categorical variables; pearson correlation coefficient, feature assessment by sliding thresholds, and signal-to-noise correlation coefficient
for continuous variables) and then select a subset of important features by a heuristic search procedure (e.g., random, genetic search) [106] [87]. It is a generic technique in machine learning and data mining for identifying the most discriminatory features in the high-dimensional data and reducing the overfitting, but its importance has been better explored for class imbalance problems that are often accompanied by the high-dimensional issue, such as text classification and bioinformatics applications [118] [112] [68] [79] [70] [119]. For example, the feature selection method significantly improved the performance of Support Vector Machine model in predicting protein function from sequence, combined with the undersampling [1].

(b) Variable Discretization

Variable discretization methods transform continuous variables into categorical variables in an unsupervised or supervised way [26]. Unsupervised techniques discretize continuous variables based on the distance or frequency without considering the target class information. Supervised techniques incorporate the target class information and determine the discretization boundaries based on statistical metrics (e.g., entropy, Gini, and the Hellinger measure) [58]. It is a generic data mining technique, but it has recently shown success in mitigating the model bias in class imbalance problems from multiple domains in a study conducted by Zhang et. al [116].

3. Algorithm

(a) Cost-Sensitive Learning

Cost-sensitive learning applies different misclassification costs to observations in the training procedure, based on their class or their attribute values, by penalizing the learning objective function (e.g., log-likelihood function) that is used for estimating models [78] [8]. To mitigate the bias caused by the fact that the observations of the minority class are less than the majority class, misclassification costs of the minority class are set to be larger than the majority class, as shown in Figure 2.2. Different from sampling methods in Figure 2.1, no data modification is
made. In Section 2.2.2, the penalized log-likelihood functions for the imbalanced data in the literature are comprehensively reviewed.

![Diagram of data and misclassification cost](image)

**Figure 2.2:** Cost-Sensitive Learning

(b) Ensemble Learning

Ensemble learning trains multiple models based on subsets of training data and aggregates the results of all models to make the final prediction to improve the accuracy by reducing the model variance and/or bias. Depending on how the training sample is selected for each model, ensemble learning is divided into the bagging-based methods [17], boosting-based methods [56], and hybrid methods [71]. The bagging approach randomly selects a subset sample of the training data, while the boosting approach selects the training sample based on the result in the previous iteration and increases the weights of observations that are misclassified. The generic ensemble learning process can be improved specifically for class imbalance problems by combining with other strategies (e.g., cost-sensitive, undersampling, oversampling) [30].

(c) Other Methods

Other methods include active learning, kernel-based learning, and one-class learning. Active learning allows the model to actively select training data yielding better performance from a large number of unlabeled data [114] [121] [35]. Kernel-based learning methods project the data into high-dimensional space and can be applied to class imbalance problems with kernel modifications adjusting the class boundary [24] [110]. One-class learning, also called novelty detection, utilizes one class of observations and is very useful for extremely imbalanced data [75] [66].
4. Output Data Level: Thresholding

Thresholding is defined to be the process of determining the decision boundary of a tunable parameter [45]. It can be applied to the probability cutoff for converting the continuous estimated probabilities to binary decisions (0 or 1) [61] [113]. It can also be applied to an internal hyperparameter of an algorithm, for example, the splitting criteria in the decision tree [45].

2.1.4 Performance Measurements

For class imbalance problems, some traditional performance measurements (e.g., accuracy) do not serve as good indicators of discrimination abilities of models [88]. He et. al suggests to provide a comprehensive assessment by combining curve-based measurements (e.g., receiver operating characteristic curve, precision-recall curve) and single-value measurements (e.g., Type I Error, Type II Error, F1 Score) [42].

To generate the single-value measurements, a choice of the threshold or probability cutoff should be made by the analysts to convert a predicted probability from the continuous model output into a binary decision for classifying whether an observation is positive or negative. The ones with the predicted probabilities smaller than the cutoff value are classified to be negative (i.e. non-event), while the ones with the predicted probabilities greater than or equal to the cutoff value are classified to be positive (i.e. event). Single-value measurements can be changed by a different cutoff, which make them less objective than the curve-based measurements. In contrast, curve-based measurements show the overall model performance under all possible probability cutoffs.

The definitions and computations of the performance measurements in the following text will be illustrated based on the example of test data in Table 2.3. It contains 10 observations with their true class in the column “True Class” and their predicted probability for the positive class by the model in the column “Predicted Probability”.
### Table 2.3: An Example of Test Data

<table>
<thead>
<tr>
<th>Observation ID</th>
<th>True Class</th>
<th>Predicted Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.09</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.15</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.34</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0.45</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0.51</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0.62</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0.78</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0.82</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0.85</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0.90</td>
</tr>
</tbody>
</table>

1. **Confusion Matrix**

A confusion matrix counts the number of observations in four classification outcomes (i.e., true positive, false positive, true negative, and false negative) by comparing the predicted class labels and the actual class labels, as shown in Table 2.4. It assumes that the minority class is the positive class coded as 1, while the majority class is the negative class coded as 0 [69].

#### Table 2.4: Confusion Matrix

<table>
<thead>
<tr>
<th></th>
<th>Actual Positive</th>
<th>Actual Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Predicted Positive</strong></td>
<td>True Positive (TP)</td>
<td>False Positive (FP)</td>
</tr>
<tr>
<td><strong>Predicted Negative</strong></td>
<td>False Negative (FN)</td>
<td>True Negative (TN)</td>
</tr>
</tbody>
</table>

All single-value measurements are calculated based on the confusion matrix. Given the outcomes in Table 2.4, the following performance measurements can be computed. True positive rate (TPR) is the proportion of actual positive observations that are correctly predicted over all actual positive observations, as defined in Eq. 2.1. It is
also called sensitivity, recall, and power. Its value ranges from 0 to 1 – the higher, the better. A high TPR is more important when it matters more to correctly identify the positive case (e.g., fraud in the fraud detection problem).

\[
\text{True Positive Rate (Sensitivity, Recall, Power)} = \frac{TP}{TP + FN} \tag{2.1}
\]

False negative rate (FNR) is the proportion of actual positive observations that are falsely predicted to be negative over all actual positive observations, as defined in Eq. 2.2. It is also called Type II Error. Its value ranges from 0 to 1 – the smaller, the better. It is an equivalent metric of TPR.

\[
\text{False Negative Rate (Type II Error)} = \frac{FN}{TP + FN} \tag{2.2}
\]

True negative rate (TNR) is the proportion of actual negative observations that are correctly predicted over all actual negative observations, as defined in Eq. 2.3. It is also called specificity. Its value ranges from 0 to 1 – the higher, the better. A high TNR is more important when it matters more not to miss a negative case (e.g., normal in the spam detection).

\[
\text{True Negative Rate (Specificity)} = \frac{TN}{FP + TN} \tag{2.3}
\]

False positive rate (FPR) is the proportion of actual negative observations that are falsely predicted to be positive over all actual negative observations, as defined in Eq. 2.4. It is also called Type I Error. Its value ranges from 0 to 1 – the smaller, the better. It is an equivalent metric of TNR.

\[
\text{False Positive Rate (Type I Error)} = \frac{FP}{FP + TN} \tag{2.4}
\]

Accuracy is the proportion of actual positive and negative observations that are correctly predicted over all observations, as defined in Eq. 2.5. Its value ranges from
0 to 1 – the higher, the better. It is a proper metric when it is equally important to classify the positive and the negative correctly.

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}
\]

Precision is the proportion of actual positive observations that are correctly predicted over all observations that are predicted to be positive, as defined in Eq. 2.6. Its value ranges from 0 to 1 – the higher, the better. It is usually considered together with the recall.

\[
\text{Precision} = \frac{TP}{TP + FP} \tag{2.6}
\]

F1 score is the harmonic mean of the precision and recall, as defined in Eq. 2.7. Its value ranges from 0 to 1 – the higher, the better. A good model should have a high precision as well as a high recall, leading to a higher F1 score.

\[
\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.7}
\]

For Table 2.3, assuming that the probability cutoff is chosen to be 0.7, the predicted class of each observation can be found in Table 2.5.
Table 2.5: Predicted Classes under Probability Cutoff 0.7

<table>
<thead>
<tr>
<th>Observation ID</th>
<th>True Class</th>
<th>Predicted Probability</th>
<th>Predicted Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.09</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.34</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0.45</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0.51</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0.62</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0.78</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0.82</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0.85</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0.90</td>
<td>1</td>
</tr>
</tbody>
</table>

By comparing the true classes and the predicted classes, we have the following outcomes.

- True Negative (TN): Observations 1, 2, 4, 5, and 6.
- False Negative (FN): Observations 3.
- True Positive (TP): Observations 7 and 9.

The resulting confusion matrix can be found in Table 2.6. The single-value performance measurements are computed and summarized in Table 2.7.

Table 2.6: Confusion Matrix of the Example Test Data

<table>
<thead>
<tr>
<th>Actual Positive</th>
<th>Actual Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Positive</td>
<td>2</td>
</tr>
<tr>
<td>Predicted Negative</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 2.7: Single-Value Performance Metrics

<table>
<thead>
<tr>
<th>Performance Metrics</th>
<th>Equation</th>
<th>Computation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
<td>( \frac{TP}{TP+FN} )</td>
<td>( \frac{2}{2+1} )</td>
<td>0.67</td>
</tr>
<tr>
<td>FNR</td>
<td>( \frac{FN}{TP+FN} )</td>
<td>( \frac{1}{2+1} )</td>
<td>0.33</td>
</tr>
<tr>
<td>TNR</td>
<td>( \frac{TN}{FP+TN} )</td>
<td>( \frac{5}{2+5} )</td>
<td>0.71</td>
</tr>
<tr>
<td>FPR</td>
<td>( \frac{FP}{FP+TN} )</td>
<td>( \frac{2}{2+5} )</td>
<td>0.29</td>
</tr>
<tr>
<td>Accuracy</td>
<td>( \frac{TP+TN}{TP+TN+FP+FN} )</td>
<td>( \frac{2+5}{2+5+2+1} )</td>
<td>0.70</td>
</tr>
<tr>
<td>Precision</td>
<td>( \frac{TP}{TP+FP} )</td>
<td>( \frac{2}{2+2} )</td>
<td>0.50</td>
</tr>
<tr>
<td>F1 Score</td>
<td>( 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} )</td>
<td>( 2 \times \frac{0.5 \times 0.667}{0.5 + 0.667} )</td>
<td>0.57</td>
</tr>
</tbody>
</table>

2. Receiver Operating Characteristic Curve (ROC Curve) and Area under ROC

A ROC curve shows how the true positive rate (i.e. sensitivity) and the false positive rate (i.e. 1—specificity) change as the probability cutoff changes. Better models achieve higher true positive rate when false positive rate is lower, which is reflected in the plot by showing ROC curve closer to the upper-left corner, leading to a larger area under the ROC curve (AUROC or AUC). Hence, a larger AUROC indicates better model performance. The ROC Curve and AUROC are well established metrics [39] [82] and not sensitive to the prior distribution of the target classes [10], making them ideal for the imbalanced data.

A ROC curve is constructed by the following steps. The resulting ROC Curve for the test data in Table 2.3 can be found in Figure 2.3.

a) Sort observations in the ascending order of their predicted probabilities for the positive class.

b) Use the predicted probability of each observation in the test data in turn as the probability cutoff. For observations with the predicted probability greater than
or equal to the probability cutoff, classify them as positive; otherwise, classify them as negative. Then count TP, FP, TN, and FN. And further compute TPR and FPR, as shown in Table 2.8.

c) Plot TPR and FPR under all probability cutoffs, as shown in Figure 2.3.

Table 2.8: Constructing ROC Curve for the Example Test Data

<table>
<thead>
<tr>
<th>Probability Cutoff</th>
<th>0.09</th>
<th>0.15</th>
<th>0.34</th>
<th>0.45</th>
<th>0.51</th>
<th>0.62</th>
<th>0.78</th>
<th>0.82</th>
<th>0.85</th>
<th>0.9</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FP</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TN</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>FN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>TPR</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.67</td>
<td>0.67</td>
<td>0.67</td>
<td>0.67</td>
<td>0.33</td>
<td>0.33</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>FPR</td>
<td>1.00</td>
<td>0.86</td>
<td>0.71</td>
<td>0.71</td>
<td>0.57</td>
<td>0.43</td>
<td>0.29</td>
<td>0.29</td>
<td>0.14</td>
<td>0.14</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Figure 2.3: ROC Curve of the Example Test Data
3. Kolmogorov-Smirnov Chart (KS Chart) and KS Statistics

A KS chart shows the difference between the cumulative percentage of positive observations and the cumulative percentage of negative observations by only considering observations in the top ranks of predicted probabilities, where the maximum difference is called KS statistics. Compared with the ROC curve, it is a more proper metric for how the models correctly classify observations with high predicted probabilities (e.g., target customers for marketing campaign).

A KS chart is constructed by the following steps.

a) Sort the observations in the descending order of their predicted probabilities for the positive class. The resorted data for Table 2.3 can be found in Table 2.9.

b) Split observations in 10 equal sized bins based on their predicted probabilities.

b) For each bin/rank, compute the number of positive observations (# positive), the number of negative observations cum (# negative), the percentage of positive observations (% positive), the percentage of negative observations (% negative), the cumulative percentage of positive observations (cum % positive), the cumulative percentage of negative observations (cum % negative), and the difference between cum % positive and cum % negative, as shown in Table 2.10.

c) Plot cum % positive and cum % negative on the y-axis and the rank on the x-axis, as shown in Figure 2.4. KS statistics is 0.38.
### Table 2.9: Observations in Descending Order of Predicted Probability for Positive Class

<table>
<thead>
<tr>
<th>Observation ID</th>
<th>Predicted Probability</th>
<th>True Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.9</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0.85</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0.82</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0.78</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0.62</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0.51</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.45</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.34</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.09</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 2.10: Constructing KS Chart for the Example Test Data

<table>
<thead>
<tr>
<th>Rank</th>
<th># Positive</th>
<th># Negative</th>
<th>% Positive</th>
<th>% Negative</th>
<th>Cum % Positive</th>
<th>Cum % Negative</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
<td>0.14</td>
<td>0.00</td>
<td>0.14</td>
<td>-0.14</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0.33</td>
<td>0.00</td>
<td>0.33</td>
<td>0.14</td>
<td>0.19</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
<td>0.14</td>
<td>0.33</td>
<td>0.29</td>
<td>0.05</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0.33</td>
<td>0.00</td>
<td>0.67</td>
<td>0.29</td>
<td>0.38</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
<td>0.14</td>
<td>0.67</td>
<td>0.43</td>
<td>0.24</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
<td>0.14</td>
<td>0.67</td>
<td>0.57</td>
<td>0.10</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
<td>0.14</td>
<td>0.67</td>
<td>0.71</td>
<td>-0.05</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0.33</td>
<td>0.00</td>
<td>1.00</td>
<td>0.71</td>
<td>0.29</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
<td>0.14</td>
<td>1.00</td>
<td>0.86</td>
<td>0.14</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
<td>0.14</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
4. Cumulative Gain and Lift Charts

Cumulative gain is the cumulative percentage of positive observations for the top percent of the sample ranked by the model, and lift is the ratio of the cumulative gain between the model and no model [34]. They evaluate the model performance on a subgroup of the sample (i.e. observations with the top ranked probabilities) instead of the whole sample, which are widely used in the applications of marketing and sales [48] [102].

They are constructed in the following steps.

a) Sort the observations in the descending order of their predicted probabilities for the positive class. The resorted data for Table 2.3 can be found in Table 2.9.

b) Split observations in 10 equal sized bins based on their predicted probabilities.

b) For each bin/rank, compute the number of positive observations ( # positive), the percentage of positive observations ( % positive), the cumulative percentage of positive observations (cum % positive), and the lift, as shown in Table 2.11.
c) For the cumulative gain chart, plot cum % positive on the y-axis and % of sample on the x-axis, as shown in Figure 2.5. For the lift chart, plot lift on the y-axis and % of sample on the x-axis, as shown in Figure 2.6.

**Table 2.11: Constructing Gain and Lift Charts for the Example Test Data**

<table>
<thead>
<tr>
<th>Rank</th>
<th>% of Sample</th>
<th># Positive</th>
<th>% Positive</th>
<th>Cum % Positive</th>
<th>Lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
<td>1</td>
<td>0.33</td>
<td>0.33</td>
<td>1.65</td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
<td>0</td>
<td>0.00</td>
<td>0.33</td>
<td>1.10</td>
</tr>
<tr>
<td>4</td>
<td>0.4</td>
<td>1</td>
<td>0.33</td>
<td>0.67</td>
<td>1.67</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>0</td>
<td>0.00</td>
<td>0.67</td>
<td>1.34</td>
</tr>
<tr>
<td>6</td>
<td>0.6</td>
<td>0</td>
<td>0.00</td>
<td>0.67</td>
<td>1.12</td>
</tr>
<tr>
<td>7</td>
<td>0.7</td>
<td>0</td>
<td>0.00</td>
<td>0.67</td>
<td>0.96</td>
</tr>
<tr>
<td>8</td>
<td>0.8</td>
<td>1</td>
<td>0.33</td>
<td>1.00</td>
<td>1.25</td>
</tr>
<tr>
<td>9</td>
<td>0.9</td>
<td>0</td>
<td>0.00</td>
<td>1.00</td>
<td>1.11</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0.00</td>
<td>1.00</td>
<td>1.0</td>
</tr>
</tbody>
</table>

**Figure 2.5: Gain Chart of the Example Test Data**
5. Probability Cutoff Choice

It is important to choose a proper probability cutoff to convert predicted probabilities from the model output to a binary decision (i.e. positive or negative). If a smaller probability cutoff is chosen, the true positive rate (i.e. sensitivity) is high but the true negative rate (i.e. specificity) is low. If a larger probability cutoff is used, the true negative rate (i.e. specificity) is high but the true positive rate (i.e. sensitivity) is low. Figure 2.7 shows how the overall accuracy, sensitivity, and specificity change as the probability cutoff increases from 0 to 1. As shown, there is some tradeoff between sensitivity and specificity. The choice of probability cutoff depends on application scenarios or problem objectives. Below are two metrics commonly used by researchers.

a) The first metric is the intersection point of true positive rate plot and true negative rate plot [38] [85]. It balances Type I Error and Type II Error. The resulting plot for the example test data in Table 2.3 is shown in Figure 2.7. Based on this
metric, the probability cutoff is 0.78. If a higher true positive rate (sensitivity) is preferred, a smaller probability cutoff can be chosen.

![Figure 2.7: Sensitivity, Specificity, and Accuracy vs. Probability Cutoff](image)

b) The other metric is the point at the maximum difference between the ROC curve and the baseline [38], or the highest Youden Index defined in Eq. 2.8. The resulting plot for the example test data in Table 2.3 is shown in Figure 2.8. Based on this metric, the probability cutoff is 0.78.

\[
\text{Youden Index} = \text{sensitivity} + \text{specificity} - 1
\]  

(2.8)
2.2 Maximum Likelihood Estimation

Maximum likelihood estimation is a method of estimating parameters of a model by maximizing a likelihood function of the observed data [80]. The following sections discuss in detail how the logistic regression and neural network models are estimated by this method.

2.2.1 Logistic Regression Model Estimation

Logistic regression is a linear model for predicting binary outcomes (e.g., fraud vs. non-fraud). In logistic regression, the values of input independent variables (i.e. $x_{i0},...,x_{in}$) are linearly combined, defined in Eq. 2.10, and then transformed by a sigmoid function, defined in Eq. 2.9, as shown in Figure 2.9.

For clarity, below are the notations used throughout the following text.

$m$ : the total number of observations in the training data

$n$ : the total number of independent variables
\( i \) : the index of observations, \( i = 1, \ldots, m \)

\( j \) : the index of independent variables, \( j = 0, \ldots, n \)

\( x_{ij} \) : the value of the \( j \)th independent variable in the \( i \)th observation

\( \mathbf{x}_i \) : the vector of values of independent variables in the \( i \)th observation

\( y_i \) : the true class label of the \( i \)th observation

\( \beta_j \) : the estimated coefficient of the \( j \)th independent variable

\( \mathbf{\beta} \) : the vector of estimated coefficients of independent variables

\( h_i \) : the model output for the \( i \)th observation

\( \hat{y}_i \) : the estimated class label for the \( i \)th observation

\[ h_i = \pi(\mathbf{\beta}^T \mathbf{x}_i) = \frac{1}{1 + e^{-\mathbf{\beta}^T \mathbf{x}_i}} \] (2.9)

where

\[ \mathbf{\beta}^T \mathbf{x}_i = \sum_{j=0}^{n} \beta_j x_{ij} = \beta_0 x_{i0} + \beta_1 x_{i1} + \ldots + \beta_n x_{in} \] (2.10)

with \( x_{i0} = 1 \), which makes \( \beta_0 \) the intercept.

The sigmoid function in Eq. 2.9 restricts the model output between 0 and 1, as shown in Figure 2.10. The model output is interpreted as the estimated probability of the event occurrence, considering that the event of the interest (e.g., fraud, delinquency, failure,
malignant) is always coded as 1 while the non-event (e.g., non-fraud, non-delinquency, pass, benign) is always coded as 0 [73] [47]. Taking the $i$th observation as an example, the probability of the event occurrence is estimated by Eq. 2.11, and correspondingly the probability of the non-event occurrence is estimated by Eq. 2.12. Mathematically, these two equations can be equivalently re-written into one equation as Eq. 2.13.

\begin{align*}
P(Y = 1|X = x_i) &= \pi(\beta^T x_i) \quad (2.11) \\
P(Y = 0|X = x_i) &= 1 - \pi(\beta^T x_i) \quad (2.12) \\
P(Y = y_i|X = x_i) &= \pi(\beta^T x_i)^{y_i}(1 - \pi(\beta^T x_i))^{(1-y_i)} \quad (2.13)
\end{align*}

Assuming that all observations are independent, the overall likelihood can be expressed by the likelihood function in Eq. 2.14, which is the product of the individual likelihood of the training data. The problem is to identify the model parameters $\beta$ that maximize the overall likelihood. To improve the computation efficiency, the likelihood function is transformed into its log form in Eq. 2.15, called log-likelihood function. To solve this unconstrained optimization problem, the most commonly used algorithm is the gradient descent algorithm [97], where the partial derivative is first computed.

\textbf{Figure 2.10:} Sigmoid Function

\begin{align*}
\pi(x) = \frac{1}{1 + e^{-x}}
\end{align*}
\[ L(\beta) = \prod_{i=1}^{m} P(Y = y_i | X = x_i) \]
\[ = \prod_{i=1}^{m} [\pi(\beta^T x_i)^{y_i} (1 - \pi(\beta^T x_i))^{(1-y_i)}] \quad (2.14) \]

\[ LL(\beta) = \sum_{i=1}^{m} [y_i \log(\pi(\beta^T x_i)) + (1 - y_i) \log(1 - \pi(\beta^T x_i))] \quad (2.15) \]

\[ \min_{\beta} - \sum_{i=1}^{m} [y_i \log(\pi(\beta^T x_i)) + (1 - y_i) \log(1 - \pi(\beta^T x_i))] \quad (2.16) \]

Maximizing the log-likelihood in Eq. 2.15 is equivalently minimizing the negative log-likelihood in Eq. 2.16, which is referred to as the loss function or cost function of logistic regression. The time complexity for solving Eq. 2.16 is \(O(n)\) by the gradient descent algorithm [28]. First, the partial derivative on \(\beta_j\) is derived in Eq. 2.17. It is updated iteratively by the rules in Eq. 2.18, where \(\alpha\) is the learning rate for \(\beta_j\). The learning rate is a hyperparameter tuned by users. The gradient descent algorithm is summarized in Algorithm 1.
\[
\frac{\partial (-LL(\beta))}{\partial \beta_j}
= - \sum_{i=1}^{m} \left[ y_i \frac{\partial \log(\pi(\beta^T x_i))}{\partial \beta_j} + (1 - y_i) \frac{\partial \log(1 - \pi(\beta^T x_i))}{\partial \beta_j} \right]
\]
\[
= - \sum_{i=1}^{m} \left[ y_i \frac{1}{\pi(\beta^T x_i)} \frac{\partial \pi(\beta^T x_i)}{\partial \beta_j} - (1 - y_i) \frac{1}{1 - \pi(\beta^T x_i)} \frac{\partial \pi(\beta^T x_i)}{\partial \beta_j} \right]
\]
\[
= - \sum_{i=1}^{m} \left[ y_i \frac{\pi(\beta^T x_i)}{\pi(\beta^T x_i) - \pi(\beta^T x_i)(1 - \pi(\beta^T x_i))} \frac{\partial \pi(\beta^T x_i)}{\partial \beta_j} \right]
\]
\[
= \sum_{i=1}^{m} \left[ \left( y_i \pi(\beta^T x_i) \right) \frac{\partial \pi(\beta^T x_i)}{\partial \beta_j} \right]
\]
\[
= \sum_{i=1}^{m} \left\{ \left[ y_i (1 - \pi(\beta^T x_i)) - (1 - y_i) \pi(\beta^T x_i) \right] \right\} x_{ij}
\]
\[
= \sum_{i=1}^{m} \left\{ \left[ y_i \pi(\beta^T x_i) - \pi(\beta^T x_i) + y_i \pi(\beta^T x_i) \right] \right\} x_{ij}
\]
\[
= \sum_{i=1}^{m} \left\{ \left[ y_i - \pi(\beta^T x_i) \right] x_{ij} \right\}
\]
\[
= \sum_{i=1}^{m} \left[ (y_i - h_i) x_{ij} \right]
\]

\[
\beta_{j,\text{NEW}} = \beta_{j,\text{CURRENT}} - \alpha \frac{\partial (-LL(\beta))}{\partial \beta_j} \]  

(2.18)
Algorithm 1: Gradient Descent Algorithm for Logistic Regression

Data: $x_i, y_i, \forall i \in [1, m]; \alpha$.

Result: $\beta_j, \forall j \in [0, n]$.

initialize $\beta_j, \forall j \in [0, n]$;

while not reaching the maximum number of iterations do

for $j \in [0, n]$ do

compute the partial derivative of $\beta_j$ in Eq. 2.17 using the current values of $\beta_j$;

update $\beta_j$ based on Eq. 2.18.

end

end

2.2.2 Penalized Log-likelihood Objective Functions for Logistic Regression

The loss function in Eq. 2.16 can be interpreted in two parts. The first part $-y_i \log(\pi(\beta^T x_i))$ is the misclassification costs for event observations (i.e. $y_i = 1$), while the second part $-(1-y_i) \log(1-\pi(\beta^T x_i))$ is the misclassification costs for non-event observations (i.e. $y_i = 0$), shown in Eq. 2.19. Because the misclassification costs are not penalized differently for events and non-events, this objective function essentially maximizes the overall accuracy.

$$cost_i = \begin{cases} 
-y_i \log(\pi(\beta^T x_i)), & \text{if } y_i = 1 \\
-(1-y_i) \log(1-\pi(\beta^T x_i)), & \text{if } y_i = 0 
\end{cases} \quad (2.19)$$

However, as Kubat et. al pointed out, the overall accuracy is not a valid and effective performance measurement for the imbalanced data [62]. In the imbalanced data, the number of observations in the majority class (i.e. non-events) is usually two times of the minority class (i.e. events) or more [43]. By maximizing the overall accuracy, logistic regression tends to be biased towards the majority class and misclassifies events as non-events severely [107] [86] [43]. For example, in an empirical study on the influence of the event rate on discrimination abilities of bankruptcy prediction models, when the event rate (i.e. the proportion of bankruptcy observations) is 0.12%, the accuracy of logistic regression model is 99.41%, but its
Type II error is 95.01%, which indicates 95.01% of bankruptcy observations are misclassified as non-bankruptcy [115]. This bias can bring great loss in practice, for example, when banks approve loans to organizations with predicted low but truly high bankruptcy probabilities. To appropriately measure the model performance on the imbalanced data, researchers have suggested to provide a comprehensive assessment with both curve-based measurements (e.g., ROC, precision-recall curve) and point-value measurements (e.g., Type I Error, Type II Error, F-measure, G-mean) [42] [62].

To apply logistic regression in the imbalanced data (i.e. rare event data), King and Zeng penalized the misclassification costs of events and non-events differently by penalty weights $W_1$ and $W_0$ in the log-likelihood function [57], as shown in Eq. 2.22. Penalty weights $W_1$ and $W_0$ are determined by the population proportion of events $\tau$ and the sample proportion of events $\bar{y}$, defined in Eq. 2.21. $W_1$ is the penalty weight for all event observations, while $W_0$ is the penalty weight for all non-event observations. Because they are invariant to values of independent variables, they are referred to as global penalty weights in this research context. Because $W_0$ and $W_1$ are pre-defined and plugged into the log-likelihood function as constants, the resulting loss function in Eq. 2.20 can be solved in the same time complexity $O(n)$, which is the same as the standard log-likelihood function in Eq. 2.16. The misclassification costs associated with this loss function can be found in Eq. 2.22. One challenge in this method is that it is hard to estimate the population proportion of events $\tau$ accurately [23], which ultimately influences the performance of logistic regression driven by global penalty weights $W_0$ and $W_1$ as found in an empirical study [116].

$$\min_{\beta} -W_1 \sum_{i=1}^{m} y_i \log(\pi(\beta^T x_i)) - W_0 \sum_{i=1}^{m} (1 - y_i) \log(1 - \pi(\beta^T x_i))$$ (2.20)

where

$$W_1 = \frac{\tau}{\bar{y}}, \quad W_0 = \frac{1 - \tau}{1 - \bar{y}}$$ (2.21)
with $\tau$ denoting the population fraction of events induced by choice-based sampling and $\bar{y}$ denoting the sample proportion of events.

$$\text{cost}_i = \begin{cases} -W_1 y_i \log(\pi(\beta^T x_i)), & \text{if } y_i = 1 \\ -W_0 (1 - y_i) \log(1 - \pi(\beta^T x_i)), & \text{if } y_i = 0 \end{cases} \quad (2.22)$$

Instead of penalizing the misclassification costs based on classes (i.e. event and non-event), Deng proposed to penalize the misclassification cost of each observation differently by a penalty weight $w_i$, where $i$ is the observation index, as shown in Eq. 2.23. $w_i$ is determined by the Gaussian kernel function, defined in Eq. 2.24, where $K_w^2$ is called the kernel width, a hyperparameter to tune. This is called the locally weighted logistic regression or kernel logistic regression [122] [15]. The resulting loss function in Eq. 2.23 can be solved in time complexity $O(n^3)$ [74] [54]. The corresponding misclassification costs of the loss function are in Eq. 2.25. The increase of the time complexity is caused by the computation of distance matrices used by the Gaussian kernel in Eq. 2.24, which limits its application on large datasets.

$$\min_{\beta} - \sum_{i=1}^{m} w_i [y_i \log(\pi(\beta^T x_i)) + (1 - y_i) \log(1 - \pi(\beta^T x_i))] \quad (2.23)$$

where

$$w_i = \exp\left(\frac{(x_i - x_q)^2}{K_w^2}\right) \quad (2.24)$$

with $x_q$ denoting the query observation being evaluated.

$$\text{cost}_i = \begin{cases} -w_i y_i \log(\pi(\beta^T x_i)), & \text{if } y_i = 1 \\ -w_i (1 - y_i) \log(1 - \pi(\beta^T x_i)), & \text{if } y_i = 0 \end{cases} \quad (2.25)$$

By including both the global penalty weights (i.e. $W_0, W_1$) and the local penalty weights (i.e. $w_i$) above along with a regularization term (i.e. $\frac{\alpha}{2}||\beta||^2$) in the log-likelihood function, Maher proposed a rare event weighted kernel logistic regression [73], as shown in Eq. 2.26. This loss function can be solved in the time complexity $O(n^3)$. The associated costs with the loss function can be found in Eq. 2.27. Besides of high computational complexity, this method also introduces one more hyperparameter $\alpha$ to tune in the regularization term.
\[
\min_{\beta} \ -W_1 \sum_{i=1}^{m} w_i y_i \log(\pi(\beta^T x_i)) - W_0 \sum_{i=1}^{m} w_i (1 - y_i) \log(1 - \pi(\beta^T x_i)) + \frac{\alpha}{2} ||\beta||^2
\]  

(2.26)

where \(\alpha\) is the regularization strength, a hyperparameter tuned by users.

\[
\text{cost}_i = \begin{cases} 
-W_1 w_i y_i \log(\pi(\beta^T x_i)) + \frac{\alpha}{2} ||\beta||^2, & \text{if } y_i = 1 \\
-W_0 w_i (1 - y_i) \log(1 - \pi(\beta^T x_i)) + \frac{\alpha}{2} ||\beta||^2, & \text{if } y_i = 0
\end{cases}
\]  

(2.27)

### 2.2.3 Neural Network Model Estimation

A neural network is a nonlinear model with three components in its typical model architecture including the input layer, the hidden layer, and the output layer. It gathers the inputs from independent variables in the input layer or upstream neurons in the hidden layer, linearly combines them, and transforms them by an activation function (e.g., sigmoid function, tanh function) \[65\]. An example model architecture can be found in Figure 2.11. As shown, the values of independent variables in the input layer are first linearly combined, defined in Eq. 2.31, and transformed by a sigmoid function, defined in Eq. 2.30, which produce values of neurons in the hidden layer. Then the values of neurons in the hidden layer are linearly combined, defined in Eq. 2.29, and transformed by a sigmoid function, defined in Eq. 2.28, which produce the output.

More notations are used to represent a neural network model, in addition to notations in Section 2.2.1.

- \(k\): the total number of neuron nodes in the hidden layer

- \(w_{1pq}\): the weight between the \(p\)th input node and the \(q\)th hidden node, \(q \in [1,k], p \in [0,n]\)

- \(w_{1q}\): the vector of all weights between all input nodes and the \(q\)th hidden node, \(q \in [1,k]\)

- \(w_{2q}\): the weight between the \(q\)th hidden node and the output node, \(q \in [1,k]\)

- \(w_2\): the vector of all weights between hidden nodes and the output node

- \(a_{iq}\): the value on the \(q\)th hidden node for the \(i\)th observation, \(q \in [1,k], i \in [1,m]\)
\( a_i \): the vector of values on all hidden nodes for the \( i \)th observation, \( i \in [1, m] \)

**Figure 2.11: Neural Network**

\[
h_i = \pi(w_2^T a_i) = \frac{1}{1 + e^{-w_2^T a_i}} \quad (2.28)
\]

where

\[
w_2^T a_i = \sum_{q=0}^{k} w_{2q} a_{iq} = w_{20} a_{i0} + w_{21} a_{i1} + \ldots + w_{2k} a_{ik} \quad (2.29)
\]

\[
a_{iq} = \pi(w_{1q}^T x_i) = \frac{1}{1 + e^{-w_{1q}^T x_i}} \quad (2.30)
\]

\[
w_{1q}^T x_i = \sum_{j=0}^{n} w_{1qj} x_{ij} = w_{1q0} x_{i0} + w_{1q1} x_{i1} + \ldots + w_{1qn} x_{in} \quad (2.31)
\]

with \( x_{i0} = 1 \) and \( a_{i0} = 1 \), which makes \( w_{110} \) and \( w_{20} \) the intercept.

The model output is generated by the sigmoid function in Eq. 2.28, ranging from 0 to 1. It is interpreted as the event occurrence probability in Eq. 2.32, assuming the event (i.e. positive class) is coded as 1 and the non-event (i.e. negative class) is coded as 0. Correspondingly, the non-event occurrence probability is the difference between 1 and the model output, denoted in Eq. 2.33. These two equations can be mathematically denoted by a single equation in Eq. 2.34.
\[ P(Y = 1|X = x_i) = \pi(w_2^T a_i) \tag{2.32} \]

\[ P(Y = 0|X = x_i) = 1 - \pi(w_2^T a_i) \tag{2.33} \]

\[ P(Y = y_i|X = x_i) = \pi(w_2^T a_i)^{y_i}(1 - \pi(w_2^T a_i))^{(1-y_i)} \tag{2.34} \]

Under the assumption of observation independence, the overall likelihood function is the product of the individual likelihood in the training data, expressed by Eq. 2.35. The problem is to identify the model parameters \( u \) that maximize the overall likelihood. To improve the computation efficiency, especially the partial derivative, its log transformation is used instead, denoted in Eq. 2.36, called log-likelihood function. This is a nonlinear nonconvex programming problem. The most efficient method is gradient descent algorithm [91] [97].

\[ L(w) = \prod_{i=1}^{m} P(Y = y_i|X = x_i) \tag{2.35} \]

\[ = \prod_{i=1}^{m} [\pi(w_2^T a_i)^{y_i}(1 - \pi(w_2^T a_i))^{(1-y_i)}] \]

\[ LL(w) = \sum_{i=1}^{m} [y_i \log(\pi(w_2^T a_i) + (1 - y_i) \log(1 - \pi(w_2^T a_i))] \tag{2.36} \]

\[ \min_{w} - \sum_{i=1}^{m} [y_i \log(\pi(w_2^T a_i) + (1 - y_i) \log(1 - \pi(w_2^T a_i))] \tag{2.37} \]

Maximizing the log-likelihood function in Eq. 2.36 is equivalently minimizing the negative log-likelihood function in Eq. 2.37, which is referred to as the loss function or cost function of the neural network. The time complexity for solving this problem is \( O(n^5) \) by using the gradient descent algorithm [29]. In the solution process, the forward propagation and the backpropagation are performed in turn in each iteration to update the values of the objective function and decision variables \( w \). In the forward propagation, the inputs are linearly combined and transformed by the activation function through the hidden layer to
produce the output (i.e. objective value). In the backpropagation, the weights $w_2$ between the output layer and the hidden layer are first updated based on their partial derive in Eq. 2.38 and learning rate $\alpha$, as shown in Eq. 2.40; then the weights $w_1$ between the hidden layer and the input layer are updated based on their partial derivative in Eq. 2.39 and learning rate $\alpha$, as shown in Eq. 2.41. The solution process is summarized in Algorithm 2.

\[
\frac{\partial (-LL(w))}{\partial w_{2q}} = -\sum_{i=1}^{m} \left[ y_i \frac{\partial \log(\pi(w_2^T a_i))}{\partial w_{2q}} + (1 - y_i) \frac{\partial \log(1 - \pi(w_2^T a_i))}{\partial w_{2q}} \right]
\]

\[
= -\sum_{i=1}^{m} \left[ y_i \frac{\partial \pi(w_2^T a_i)}{\partial w_{2q}} - (1 - y_i) \frac{\partial \pi(w_2^T a_i)}{1 - \pi(w_2^T a_i)} \right] \frac{\partial (w_2^T a_i)}{\partial w_{2q}}
\]

\[
= -\sum_{i=1}^{m} \left[ \left( \frac{y_i}{\pi(w_2^T a_i)} - \frac{(1 - y_i)}{1 - \pi(w_2^T a_i)} \right) \pi(w_2^T a_i)(1 - \pi(w_2^T a_i)) \frac{\partial (w_2^T a_i)}{\partial w_{2q}} \right]
\]

\[
= -\sum_{i=1}^{m} \left[ \left( \frac{y_i}{\pi(w_2^T a_i)} - \frac{(1 - y_i)}{1 - \pi(w_2^T a_i)} \right) \pi(w_2^T a_i)(1 - \pi(w_2^T a_i))(1 - \pi(w_2^T a_i))a_{iq} \right]
\]

\[
= -\sum_{i=1}^{m} \left[ \frac{y_i(1 - \pi(w_2^T a_i)) - (1 - y_i)\pi(w_2^T a_i)}{\pi(w_2^T a_i)(1 - \pi(w_2^T a_i))} \right] \pi(w_2^T a_i)(1 - \pi(w_2^T a_i))a_{iq}
\]

\[
= -\sum_{i=1}^{m} \left\{ [(y_i(1 - \pi(w_2^T a_i)) - (1 - y_i)\pi(w_2^T a_i)]a_{iq} \right\}
\]

\[
= -\sum_{i=1}^{m} \left\{ [y_i - y_i\pi(w_2^T a_i) - \pi(w_2^T a_i) + y_i\pi(w_2^T a_i)]a_{iq} \right\}
\]

\[
= -\sum_{i=1}^{m} \left\{ [y_i - \pi(w_2^T a_i)]a_{iq} \right\}
\]

\[
= -\sum_{i=1}^{m} [(y_i - h_i)a_{iq}]
\]
$$\frac{\partial (-LL(w))}{\partial w_{1q}} = \frac{\partial (-LL(w))}{\partial a_{iq}} \frac{\partial a_{iq}}{\partial w_{1q}}$$

$$= \sum_{i=1}^{m} \left\{ [y_i - \pi(w_2^T a_i)] w_{2q} \frac{\partial \pi(w_{1q}^T x_i)}{\partial w_{1q}} \right\}$$

$$= \sum_{i=1}^{m} \left\{ [y_i - \pi(w_2^T a_i)] w_{2q} \pi(w_{1q}^T x_i)(1 - \pi(w_{1q}^T x_i)) \frac{\partial (w_{1q}^T x_i)}{\partial w_{1q}} \right\}$$

$$= \sum_{i=1}^{m} \left\{ [y_i - \pi(w_2^T a_i)] w_{2q} \pi(w_{1q}^T x_i)(1 - \pi(w_{1q}^T x_i)) x_{ij} \right\}$$

$$= \sum_{i=1}^{m} [(y_i - h_i) w_{2q} a_{iq} (1 - a_{iq}) x_{ij}]$$

$$w_{2q, NEW} = w_{2q, CURRENT} - \alpha \frac{\partial (-LL(w))}{\partial w_{2q, CURRENT}}$$

$$w_{1q, NEW} = w_{1q, CURRENT} - \alpha \frac{\partial (-LL(w))}{\partial w_{1q, CURRENT}}$$
Algorithm 2: Gradient Descent Algorithm for Neural Network

Data: $x_i, y_i, \forall i \in [1, m]; \alpha$

Result: $w_{2q}; w_{1qp}, \forall q \in [1, k], p \in [0, n]$.

initialize $w_{2q}; w_{1qp}, \forall q \in [1, k], p \in [0, n]$;

while not reaching the maximum number of iterations do
  for $j \in [0, n]$ do
    compute the linear combination and activation value from the input layer to
    the hidden layer based on Eq. 2.31 and Eq. 2.30;
    compute the linear combination and activation value from the hidden layer to
    the output layer based on Eq. 2.29 and Eq. 2.28;
    compute the partial derivative of $w_{2q}$ based on Eq. 2.38;
    update $w_{2q}$ based on Eq. 2.40;
    compute the partial derivative of $w_{1qp}$ based on Eq. 2.39;
    update $w_{1qp}$ based on Eq. 2.41.
  end
end

2.2.4 Penalized Log-likelihood Objective Functions for Neural Network

Similar to logistic regression in Section 2.2.2, the loss function of neural networks in Eq. 2.37 can be interpreted in two parts. The first part $-y_i \log(\pi(w_2^T a_i))$ is the misclassification costs for event observations (i.e. $y_i = 1$), while the second part $-(1 - y_i) \log(1 - \pi(w_2^T a_i))$ is the misclassification costs for non-event observations (i.e. $y_i = 0$), shown in Eq. 2.42.

$$cost_i = \begin{cases} 
  -y_i \log(\pi(w_2^T a_i)), & \text{if } y_i = 1 \\
  -(1 - y_i) \log(1 - \pi(w_2^T a_i)), & \text{if } y_i = 0 
\end{cases} \quad (2.42)$$

Different from logistic regression, for neural networks, misclassification costs are usually penalized by a constant cost matrix in Table 2.12, which is also referred to as cost-sensitive neural network [63]. The cost matrix is derived from the problem context by domain experts.
and known in the modeling process [69]. After incorporating the cost matrix in the log-likelihood function, the loss function becomes Eq. 2.43, where the misclassification costs for events and non-events are penalized by the cost of false positive ($C_{FP}$) and false negative ($C_{FN}$), respectively, as shown in Eq. 2.44. Because the values in the cost matrix are plugged into the function as constants, the time complexity of solving Eq. 2.43 is the same as the standard neural network in Eq. 2.37, $O(n^5)$.

<table>
<thead>
<tr>
<th>Table 2.12: Cost Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition Positive</td>
</tr>
<tr>
<td>Predicted Positive</td>
</tr>
<tr>
<td>Predicted Negative</td>
</tr>
</tbody>
</table>

$$\min_w - \sum_{i=1}^{m} [C_{FP}y_i \log(\pi(w_2^T a_i)) + C_{FN}(1 - y_i) \log(1 - \pi(w_2^T a_i))] \quad (2.43)$$

$$cost_i = \begin{cases} 
-C_{FP}y_i \log(\pi(w_2^T a_i)), & \text{if } y_i = 1 \\
-C_{FN}(1 - y_i) \log(1 - \pi(w_2^T a_i)), & \text{if } y_i = 0 
\end{cases} \quad (2.44)$$

## 2.3 Model Evaluation

Models are generally evaluated on different training data and test data. To generate multiple sets of training and test samples, two resampling strategies are commonly used, including K-fold cross validation and bootstrapping [55] [12]. Furthermore, models are diagnosed by decomposing model errors into the bias and variance.

### 2.3.1 Resampling Strategies

In the K-fold cross validation, the data is first split into K folds. Each fold is used as the validation data in turn, and all the other folds are used as the training dataset. Figure 2.12 shows 10-fold cross validation process, where the training data is highlighted in white and the validation data is highlighted in black.
In the bootstrapping, the dataset is first split into training data and validation data. A different bootstrap training sample is selected from the initial training data with replacement and used as the training data in each iteration, as shown in Figure 2.13.
By their nature, K-fold cross validation reflects more the model generalization ability on the new data, while bootstrapping emphasizes more the significance of model coefficients. In the following section, K-fold cross validation is used to measure machine learning bias and variance, while bootstrapping is used to measure statistical bias and variance, for accommodating different definitions of the bias in machine learning and statistics [22]. The variance refers to the amount of the change of the estimated model if using different training data [50], which has the same definition in machine learning and statistics despite different measurements.

2.3.2 Machine Learning Bias and Variance

Machine learning bias is defined to be “any basis for choosing one generalization over another, other than strict consistency with the observed training instances” [77]. It emphasizes the generalization ability of learning algorithms and shows how well a learning algorithm performs on new data. For that purpose, the repeated 10-fold cross validation is used to resample the data to generate different training data and validation data in each run. Figure 2.12 shows one run of 10-fold cross validation which contain 10 iterations.

The computational detail of the machine learning bias and variance can be found below.

\begin{align*}
    r &: \text{the total number of repeated runs of 10-fold cross validation;} \\
    w &: \text{the run index of 10-fold cross validation, } w = 1, ..., r; \\
    v &: \text{the iteration index in one run of 10-fold cross validation, } v = 1, ..., 10; \\
    y_{wuv} &: \text{the true class label of the } u\text{th observation in the } v\text{th iteration of the } w\text{th cross validation run;} \\
    x_{wuv} &: \text{the vector of values of independent variables in the } u\text{th observation in the } v\text{th iteration of the } w\text{th cross validation run;} \\
\end{align*}

1. In each run of 10-fold cross validation, denoted as \( w \), shuffle the data with a different random seed and then divide the shuffled data into 10 folds in a stratified manner based on the distribution of the dependent variable.
2. For the $v$th iteration in the $w$th run of the cross validation,

(a) train a model with the data in the training folds;
(b) obtain the model coefficient vector $\hat{\beta}_{wv}^T$;
(c) estimate the probability $h_{wvu}$ of each observation in the validation fold;

\[ h_{wvu} = \pi(\hat{\beta}_{wv}^T x_{wvu}) \]  

(2.45)

(d) compute the AUROC of the model, denoted as $AUROC_{wv}$, based on $h_{wv}$ and $y_{wv}$.

3. Compute the mean of the AUROCs across all runs of 10-fold cross validation.

\[ N = r \times 10 \]  

(2.46)

\[ \mu = \frac{\sum_{w=1}^{r} \sum_{v=1}^{10} AUROC_{wv}}{N} \]  

(2.47)

4. Compute the standard deviation of the AUROCs across all runs of 10-fold cross validation.

\[ \sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (AUROC_{wv} - \mu)^2} \]  

(2.48)

5. Compute the 95% confidence interval of the AUROCs across all runs of 10-fold cross validation.

\[ \left( \mu - z^* \frac{\sigma}{\sqrt{N}}, \mu + z^* \frac{\sigma}{\sqrt{N}} \right) \]  

(2.49)

where $z^* = 1.96$.

### 2.3.3 Statistical Bias and Variance

Statistical bias is defined as “the persistent or systematic error that a learning algorithm is expected to make when trained on training sets of size $m$” [22]. For that, the bootstrapping is used to generate multiple training datasets. The statistical bias and variance are decomposed
from model errors. They are originally defined for the squared loss in regression problems and extended for 0-1 loss in classification problems, as shown in Table 2.13.

The definitions for the squared loss are as follows.

- The loss is the squared difference between the true class label $y$ and the estimated probability $h$.
- The main prediction is the mean of estimated probabilities $E(h)$.
- The bias is the difference between the true class label $y$ and the mean of estimated probabilities $E(h)$.
- The variance is the mean of the squared difference between the mean of estimated probabilities $E(h)$ and a specific estimated probability $h$.

Comparatively, the extended definitions for the 0-1 loss can be found below.

- The loss is defined to be 1 if the true class label $y$ is different from the predicted class label $\hat{y}$ and 0 otherwise.
- The main prediction is the mode or the most frequently predicted class label $E(\hat{y})$.
- The bias is the difference between the true class label $y$ and the mode of predicted class labels $E(\hat{y})$.
- The variance is the mean of the difference between the mode of predicted class labels $E(\hat{y})$ and a specific predicted class label $\hat{y}$.

<table>
<thead>
<tr>
<th></th>
<th>Squared Loss</th>
<th>0-1 Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Loss</td>
<td>$(y - h)^2$</td>
<td>$L(y, \hat{y})$</td>
</tr>
<tr>
<td>Expected Loss</td>
<td>$E((y - h)^2)$</td>
<td>$E(L(y, \hat{y}))$</td>
</tr>
<tr>
<td>Main Prediction $E(h)$ or $E(\hat{y})$</td>
<td>mean</td>
<td>mode</td>
</tr>
<tr>
<td>$Bias^2$</td>
<td>$(y - E(h))^2$</td>
<td>$L(y, E(\hat{y}))$</td>
</tr>
<tr>
<td>$Variance$</td>
<td>$E[(E(h) - h)^2]$</td>
<td>$E[L(E(\hat{y}), \hat{y})]$</td>
</tr>
</tbody>
</table>

Table 2.13: Bias-Variance Decompositions
\[ L(y, \hat{y}) = \begin{cases} 
1, & \text{if } y \neq \hat{y} \\
0, & \text{Otherwise} 
\end{cases} \] (2.50)

The computational detail of the bias-variance decompositions can be found below.

- \( s \): the total number of iterations of bootstrapping
- \( t \): the total number of observations in the validation data
- \( k \): the iteration index of bootstrapping, \( k = 1, \ldots, s \)
- \( l \): the index of observations in the validation data, \( l = 1, \ldots, t \)
- \( b_k \): the training data in the \( k \)th iteration
- \( y_l \): the true class label of the \( l \)th observation in the validation data
- \( x_l \): the vector of values of independent variables in the \( l \)th observation in the validation data

1. Divide the dataset into the training data and validation data.

2. For each iteration, generate a training dataset with replacement by a different random seed from the training data in Step 1, as shown in Figure 2.13.

3. Based on the training data \( b_k \) in each iteration,
   - (a) train a model;
   - (b) obtain the model coefficient vector \( \hat{\beta}_k^{T} \);
   - (c) estimate the probability \( h_{kl} \) of each observation in the validation data, \( \forall l = 1, \ldots, t \);
     \[ h_{kl} = \pi(\hat{\beta}_k^{T} x_l) \] (2.51)
   - (d) compare the estimated probability \( h_{kl} \) and the probability cutoff;
   - (e) determine the predicted label \( \hat{y}_{kl} \): if \( h_{kl} \) is greater than the probability cutoff, set \( \hat{y}_{kl} \) to be 1; otherwise, set \( \hat{y}_{kl} \) to be 0.
4. Compute the mean \( E(h_l) \) of the estimated probabilities \( h_{kl} \) of the \( l \)th observation in the validation data by all bootstrap models in Step 3.

\[
E(h_l) = \frac{\sum_{k=1}^{s} h_{kl}}{s} \tag{2.52}
\]

5. Compute the bias and variance decomposition of the squared loss (i.e. \( SL \)) [25].

\[
squared \text{ loss} = \sum_{k=1}^{s} \sum_{l=1}^{t} (h_{kl} - y_l)^2 \tag{2.53}
\]

\[
\text{bias}^2 = \frac{\sum_{l=1}^{t} (E(h_l) - y_l)^2}{t} \tag{2.54}
\]

\[
\text{variance} = \frac{\sum_{k=1}^{s} \sum_{l=1}^{t} (h_{kl} - E(h_l))^2}{s \times t - 1} \tag{2.55}
\]

6. Compute the mode \( E(\hat{y}_l) \) of the predicted class labels \( \hat{y}_{kl} \) of the \( l \)th observation in the validation data by all bootstrap models in Step 3. If \( \frac{\sum_{k=1}^{s} \hat{y}_{kl}}{s} > 0.5 \), then \( E(\hat{y}_l) = 1 \); otherwise, \( E(\hat{y}_l) = 0 \).

7. Compute the bias and variance decomposition of the 0–1 classification loss (i.e. \( CL \)) [25].

\[
\text{classification loss} = \sum_{k=1}^{s} \sum_{l=1}^{t} (\hat{y}_{kl} - y_l) \tag{2.56}
\]

\[
\text{bias}^2 = P(E(\hat{y}_l) \neq y_l) = \frac{\sum_{l=1}^{t} B_l}{t} \tag{2.57}
\]

\[
\text{variance} = P(\hat{y}_{kl} \neq E(\hat{y}_l)) = \frac{\sum_{k=1}^{s} \sum_{l=1}^{t} V_{kl}}{s \times t - 1} \tag{2.58}
\]

where

\[
B_l = \begin{cases} 
1, & \text{if } E(\hat{y}_l) \neq y_l \\
0, & \text{Otherwise}
\end{cases} \tag{2.59}
\]
\[ V_{kl} = \begin{cases} 
1, & \text{if } \hat{y}_{kl} \neq E(\hat{y}) \\
0, & \text{Otherwise} 
\end{cases} \] (2.60)

As shown in Step 4 and Step 6 above, the main prediction of multiple bootstrap models for one observation is defined differently for the squared loss and the 0-1 loss. In the squared loss, the main prediction is the mean of the estimated probabilities, while in the 0-1 loss, the main prediction is the mode of the predicted class labels \[25\]. The intent is to adapt the bias-variance decomposition from the squared loss for regression models to the 0-1 loss for classification models to help better understand the model learning abilities.
Chapter 3

A Novel Penalized Log-likelihood Objective Function

To address the challenges of hard parameter estimating (i.e. population proportion of the event) and high time complexity in the related work, we propose a novel penalized log-likelihood objective function by including penalty weights for event observations as decision variables and learning them from data along with model coefficients via the gradient descent algorithm.

3.1 Definition for Logistic Regression

By introducing the local penalty weights $\lambda_i$ for event observations as decision variables in the log-likelihood objective function, the loss function is redefined in Eq. 3.1, denoted as $-LL(\beta, \lambda)$. The misclassification costs of event observations are penalized by $\lambda_i$, while the misclassification costs of non-event observations are not penalized, shown in Eq. 3.2. The intention is to reduce the number of decision variables and the complexity of the optimization problem.

$$
\min_{\beta, \lambda} \sum_{i=1}^{m} \left[ \lambda_i y_i \log(\pi(\beta^T x_i)) + (1 - y_i) \log(1 - \pi(\beta^T x_i)) \right] 
$$  
(3.1)
where $\lambda_i > 0$, a parameter learned from the data.

$$
cost_i = \begin{cases} 
-\lambda_i y_i \log(\pi(\beta^T x_i)), & \text{if } y_i = 1 \\
-(1 - y_i) \log(1 - \pi(\beta^T x_i)), & \text{if } y_i = 0 
\end{cases} \tag{3.2}
$$

### 3.2 Learning by Gradient Descent

The optimization of the proposed log-likelihood function in Eq. 3.1 is a nonlinear programming problem with two sets of decision variables $\beta$ and $\lambda$, which can be solved by the gradient descent algorithm in the time complexity $O(n)$. First, the partial derivative on $\beta_j$ and $\lambda_i$ is derived in Eq. 3.3 and Eq. 3.4, respectively. They are updated iteratively by the rules in Eq. 3.5 and Eq. 3.6 correspondingly, where $\alpha_1$ is the learning rate for $\beta_j$ and $\alpha_2$ is the learning rate for $\lambda_i$. The learning rates are hyperparameters tuned by users. The modified gradient descent algorithm is summarized in Algorithm 3.
\[
\frac{\partial(-LL(\beta, \lambda))}{\partial \beta_j} = -\sum_{i=1}^{m}[\lambda_i y_i \frac{\partial \log(\pi(\beta^T x_i))}{\partial \beta_j} + (1 - y_i) \frac{\partial \log(1 - \pi(\beta^T x_i))}{\partial \beta_j}]
\]
\[
= -\sum_{i=1}^{m}\frac{\lambda_i y_i}{\pi(\beta^T x_i)} \frac{\partial \pi(\beta^T x_i)}{\partial \beta_j} - (1 - y_i) \frac{1}{1 - \pi(\beta^T x_i)} \frac{\partial \pi(\beta^T x_i)}{\partial \beta_j}
\]
\[
= -\sum_{i=1}^{m}\left[\left(\frac{\lambda_i y_i}{\pi(\beta^T x_i)} - \frac{1 - y_i}{1 - \pi(\beta^T x_i)}\right) \frac{\partial \pi(\beta^T x_i)}{\partial \beta_j}\right]
\]
\[
= \sum_{i=1}^{m}\left[\left(\frac{\lambda_i y_i}{\pi(\beta^T x_i)} - \frac{1 - y_i}{1 - \pi(\beta^T x_i)}\right) \frac{\partial(\beta^T x_i)}{\partial \beta_j}\right]x_{ij} (3.3)
\]
\[
= \sum_{i=1}^{m}\{[\lambda_i y_i(1 - \pi(\beta^T x_i)) - (1 - y_i)\pi(\beta^T x_i)]x_{ij}\}
\]
\[
= \sum_{i=1}^{m}\{[\lambda_i y_i - \lambda_i y_i \pi(\beta^T x_i) - \pi(\beta^T x_i) + y_i \pi(\beta^T x_i)]x_{ij}\}
\]
\[
= \sum_{i=1}^{m}[(\lambda_i y_i - \lambda_i y_i h_i - h_i + y_i h_i) x_{ij}]
\]
\[
\frac{\partial(-LL(\beta, \lambda))}{\partial \lambda_i} = y_i \log(\pi(\beta^T x_i)) (3.4)
\]
\[
\beta_{j,NEW} = \beta_{j,CURRENT} - \alpha_1 \frac{\partial(-LL(\beta, \lambda))}{\partial \beta_j,CURRENT} (3.5)
\]
\[
\lambda_{i,NEW} = \lambda_{i,CURRENT} - \alpha_2 \frac{\partial(-LL(\beta, \lambda))}{\partial \lambda_i,CURRENT} (3.6)
\]
Algorithm 3: Modified Gradient Descent Algorithm for Logistic Regression

Data: \( x_i, y_i, \forall i \in [1, m]; \alpha_1; \alpha_2 \).

Result: \( \beta_j, \forall j \in [0, n] \).

initialize \( \beta_j, \forall j \in [0, n]; \lambda_i, \forall i \in [1, m] \);

\[ \text{while not reaching the maximum number of iterations do} \]
\[ \text{for } j \in [0, n] \text{ do} \]
\[ \quad \text{compute the partial derivative of } \beta_j \text{ in Eq. 3.3 using the current values of } \beta_j \]
\[ \quad \text{and } \lambda_i; \]
\[ \quad \text{update } \beta_j \text{ based on Eq. 3.5.} \]
\[ \text{end} \]
\[ \text{for } i \in [1, m] \text{ do} \]
\[ \quad \text{compute the partial derivative of } \lambda_i \text{ in Eq. 3.4 using the current values of } \beta_j; \]
\[ \quad \text{update } \lambda_i \text{ based on Eq. 3.6.} \]
\[ \text{end} \]
\[ \text{end} \]

3.3 Probability Estimating

To interpret the role of local penalty weights \( \lambda \), we reverse the log in Eq. 3.1 and trace it back to the likelihood function. As shown in Eq. 3.7, the penalty weights \( \lambda \) essentially regularize the process of learning the model coefficients \( \beta \) from the training data by weighting the estimated probabilities. Then the learned \( \beta \) are used to estimate the probability for the event occurrence based on Eq. 2.11 on the validation data and test data. Because \( \lambda \) only regularizes the learning process and is not used for the probability estimation together with \( \beta \), the interpretability of logistic regression is maintained.

\[
L(\beta) = \prod_{i=1}^{m} P(Y = y_i | X = x_i)^{\lambda_i} = \prod_{i=1}^{m} \left[ \pi (\beta^T x_i)^{y_i} (1 - \pi (\beta^T x_i))^{(1-y_i)} \right]^{\lambda_i} \tag{3.7}
\]

where \( \lambda_i = 1 \) when \( y_i = 0 \) and \( \lambda_i \) are values learned from data when \( y_i = 1 \).
3.4 Comparison with Other Penalized Log-likelihood Functions

Our proposed penalized log-likelihood function is compared comprehensively with the existing log-likelihood functions in Table 3.1. As a linear model, it considers the imbalance of the data, is much less complicated in perspectives of time complexity and the number of estimated parameter sets than nonlinear models with penalty weights determined by Gaussian kernel, and does not have any penalty weight-related hyperparameter to tune. Its advantages will be demonstrated in experimental results in Section 4.2.

Table 3.1: Comparison of Penalized Log-likelihood Functions

<table>
<thead>
<tr>
<th>Equation</th>
<th>Imbalance</th>
<th>Linearity</th>
<th>Computational Complexity</th>
<th>Estimated Parameter Sets</th>
<th>Hyperparameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eq. 2.16</td>
<td>No</td>
<td>linear</td>
<td>$O(n)$</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>Eq. 2.20</td>
<td>Yes</td>
<td>linear</td>
<td>$O(n)$</td>
<td>1</td>
<td>$\tau$</td>
</tr>
<tr>
<td>Eq. 2.23</td>
<td>Yes</td>
<td>nonlinear</td>
<td>$O(n^3)$</td>
<td>Test Size</td>
<td>$K_w$</td>
</tr>
<tr>
<td>Eq. 2.26</td>
<td>Yes</td>
<td>nonlinear</td>
<td>$O(n^3)$</td>
<td>Test Size</td>
<td>$\tau, K_w$</td>
</tr>
<tr>
<td>Eq. 3.1</td>
<td>Yes</td>
<td>linear</td>
<td>$O(n)$</td>
<td>1</td>
<td>N/A</td>
</tr>
</tbody>
</table>

3.5 Extension for Neural Network

For neural networks, to learn the local penalty weights $\lambda$ for event observations from data along with the model parameters $w$, its optimization objective is extended to be Eq. 3.8, denoted as $-LL(w, \lambda)$. And the misclassification costs become Eq. 3.9.

$$\min_{w, \lambda} - \sum_{i=1}^{m} [\lambda_i y_i \log(\pi(w^T_2 a_i) + (1 - y_i) \log(1 - \pi(w^T_2 a_i))]$$

(3.8)

where $\lambda_i > 0$, a parameter learned from the data.

$$cost_i = \begin{cases} 
-\lambda_i y_i \log(\pi(w^T_2 a_i)), & \text{if } y_i = 1 \\
-(1 - y_i) \log(1 - \pi(w^T_2 a_i)), & \text{if } y_i = 0 
\end{cases}$$

(3.9)
This is a nonlinear optimization problem on decision variables \( w \) and \( \lambda \), which can be solved by the gradient descent algorithm in the same time complexity \( O(n^5) \) as the standard log-likelihood function for neural networks in Eq. 2.37. For decision variables \( w \) and \( \lambda \), their partial derivatives are first derived in Eq. 3.10, 3.11, and 3.12. And they are updated iteratively based on their learning rates in Eq. 3.10, 3.11, and 3.15. The solution process is summarized in Algorithm 4.

\[
\frac{\partial (-LL(w, \lambda))}{\partial w_{2q}} = - \sum_{i=1}^{m} \left[ \lambda_i y_i \frac{\partial \log(\pi(w_2^T a_i))}{\partial w_{2q}} + (1 - y_i) \frac{\partial \log(1 - \pi(w_2^T a_i))}{\partial w_{2q}} \right]
\]

\[
= - \sum_{i=1}^{m} \left[ \frac{\lambda_i y_i}{\pi(w_2^T a_i)} - (1 - y_i) \frac{1}{1 - \pi(w_2^T a_i)} \right] \frac{\partial \pi(w_2^T a_i)}{\partial w_{2q}}
\]

\[
= - \sum_{i=1}^{m} \left[ \frac{\lambda_i y_i}{\pi(w_2^T a_i)} - (1 - y_i) \frac{1}{1 - \pi(w_2^T a_i)} \right] \pi(w_2^T a_i)(1 - \pi(w_2^T a_i)) \frac{\partial w_2^T a_i}{\partial w_{2q}}
\]

\[
= - \sum_{i=1}^{m} \left[ \frac{\lambda_i y_i (1 - \pi(w_2^T a_i)) - (1 - y_i) \pi(w_2^T a_i)}{\pi(w_2^T a_i)(1 - \pi(w_2^T a_i))} \right] \pi(w_2^T a_i)(1 - \pi(w_2^T a_i)) a_{iq}
\]

\[
= - \sum_{i=1}^{m} \left[ \frac{\lambda_i y_i (1 - \pi(w_2^T a_i)) - (1 - y_i) \pi(w_2^T a_i)}{\pi(w_2^T a_i)(1 - \pi(w_2^T a_i))} \right] \pi(w_2^T a_i)(1 - \pi(w_2^T a_i)) a_{iq}
\]

\[
= - \sum_{i=1}^{m} \left[ \lambda_i y_i h_i - h_i + y_i h_i \right] a_{iq}
\]
\[
\frac{\partial(-LL(w, \lambda))}{\partial w_{1qp}} = \frac{\partial(-LL(w))}{\partial a_{iq}} \frac{\partial a_{iq}}{\partial w_{1qp}}
\]

\[
= -\sum_{i=1}^{m} \left\{ [y_i - \pi(w_2^T a_i)]w_{2q} \pi(w_1^T x_i)(1 - \pi(w_1^T x_i)) \frac{\partial(w_1^T x_i)}{\partial w_{1qp}} \right\}
\]

\[
= -\sum_{i=1}^{m} \left\{ [y_i - \pi(w_2^T a_i)]w_{2q} \pi(w_1^T x_i)(1 - \pi(w_1^T x_i))x_{ij} \right\}
\]

\[
= -\sum_{i=1}^{m} \left\{ [(y_i - h_i)w_{2q}a_{iq}(1 - a_{iq})x_{ij}] \right\}
\]

\[
\frac{\partial(-LL(w, \lambda))}{\partial \lambda_i} = y_i \log(\pi(w_2^T a_{iq}))
\] (3.12)

\[
w_{2q,NEW} = w_{2q,CURRENT} - \alpha_1 \frac{\partial(-LL(w, \lambda))}{\partial w_{2q,CURRENT}}
\] (3.13)

\[
w_{1qp,NEW} = w_{1qp,CURRENT} - \alpha_1 \frac{\partial(-LL(w, \lambda))}{\partial w_{1qp,CURRENT}}
\] (3.14)

\[
\lambda_{i,NEW} = \lambda_{i,CURRENT} - \alpha_2 \frac{\partial(-LL(w, \lambda))}{\partial \lambda_i,CURRENT}
\] (3.15)
Algorithm 4: Modified Gradient Descent Algorithm for Neural Network

Data: $x_i, y_i, \forall i \in [1, m]; \alpha$.

Result: $w_{2q}; w_{1qp}, \forall q \in [1, k], p \in [0, n]$.

initialize $w_{2q}; w_{1qp}, \forall q \in [1, k], p \in [0, n]; \lambda_i, \forall i \in [1, m]$.

while not reaching the maximum number of iterations do
  for $j \in [0, n]$ do
    compute the linear combination and activation value from the input layer to
    the hidden layer based on Eq. 2.31 and Eq. 2.30;
    compute the linear combination and activation value from the hidden layer to
    the output layer based on Eq. 2.29 and Eq. 2.28;
    compute the partial derivative of $w_{2q}$ based on Eq. 3.10;
    update $w_{2q}$ based on Eq. 3.13;
    compute the partial derivative of $w_{1qp}$ based on Eq. 3.11;
    update $w_{1qp}$ based on Eq. 3.14.
  end
  for $i \in [1, m]$ do
    compute the partial derivative of $\lambda_i$ in Eq. 3.12;
    update $\lambda_i$ based on Eq. 3.15.
  end
end
Chapter 4

Improving Logistic Regression

4.1 Introduction

Logistic regression has been widely used for decision making since its development by Cox and Duncan independently in the 1950s and 1960s for binary classification problems [18][19][104]. It is a linear classification model estimating the probability of an event occurrence (e.g., fraud, delinquency, failure, malignant) [72]. The advantage of logistic regression is multifold, including high interpretability and low time complexity [47]. The application of logistic regression covers broad areas, such as bankruptcy prediction[64], credit scoring [8], heart sound segmentation [96], landslide susceptibility prediction [20], urban land spatial expansion analysis [94] and adolescent obesity risk [117].

However, when trained on the imbalanced data, logistic regression does not differentiate events and non-events effectively [107]. For example, in an empirical study on the influence of the event rate on discrimination abilities of bankruptcy prediction models, when the event rate (i.e. the proportion of bankruptcy observations) is 0.12%, the accuracy of logistic regression model is 99.41%, but its Type II error is 95.01%, which indicates 95.01% of bankruptcy observations are misclassified as non-bankruptcy [115]. This bias can bring great loss in practice, for example, when banks approve loans to organizations with low predicted but actually high bankruptcy probabilities.

The bias is caused by the underlying assumption of the optimization objective (i.e. log-likelihood function), which is to maximize the overall accuracy [86] [43]. However, the overall
accuracy is not a valid performance measurement for the classification of imbalanced data [62].

One strategy for mitigating this bias is to penalize the misclassification costs of observations differently in the log-likelihood objective function by either global or local penalty weights when training the logistic regression model for optimal coefficients by the maximum likelihood method [103]. The global penalty weights are determined by the population proportion and the sample proportion of events, so they are different for the event class and the non-event class but the same for observations in the same class, invariant to values of independent variables [57]. The local penalty weights are determined by the Gaussian kernel, making them unique for each observation [21]. However, for the global penalty weights, it is challenging to accurately estimate the population proportion of events, which ultimately influences the performance of logistic regression, since it is found that logistic regression performs differently on the same data when penalized by different global penalty weights in an empirical study [116]. On the other hand, logistic regression with local penalty weights, referred to as kernel logistic regression, has the time complexity $O(n^3)$ [74], limiting its application on large datasets.

In the present work, we apply the proposed penalized log-likelihood objective function in Chapter 3 to train logistic regression models, where the penalty weights for event observations are included as decision variables and learned from data along with model coefficients via the gradient descent method. Our experimental results demonstrate that both differentiation ability and computation efficiency of logistic regression can be improved on the imbalanced data with this method.

This chapter is structured as follows. Section 4.2 illustrates the experiments and results. In Section 4.3, the estimated probability distributions and the estimated coefficients of models are examined on a credit dataset as a case study. In Section 4.4, the statistical bias and variance of models are analyzed. In Section 4.5, the conclusions are presented.
4.2 Experiments

Imbalanced data exists in almost every area. 10 public imbalanced datasets from multiple domains are collected and used in the experimental study of the following chapters. Their basic characteristics can be found in Table 4.1, including data source, target (i.e. dependent variable), event rate, the number of observations, the number of variables, variable types, and domain area. The event rate in these datasets ranges from 0.76% to 10.42%.

Table 4.1: Basic Characteristics of Datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Repository</th>
<th>Target</th>
<th>Event Rate</th>
<th>Observations</th>
<th>Attributes</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>abalone_19</td>
<td>UCI</td>
<td>19</td>
<td>0.76%</td>
<td>4177</td>
<td>7C,1N</td>
<td>Life</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>UCI</td>
<td>06</td>
<td>5.55%</td>
<td>452</td>
<td>206C, 73N</td>
<td>Biology</td>
</tr>
<tr>
<td>ecoli</td>
<td>UCI</td>
<td>imU</td>
<td>10.42%</td>
<td>336</td>
<td>7C</td>
<td>Life</td>
</tr>
<tr>
<td>oil</td>
<td>UCI</td>
<td>minority</td>
<td>4.35%</td>
<td>937</td>
<td>49C</td>
<td>Environment</td>
</tr>
<tr>
<td>ozone_level</td>
<td>UCI</td>
<td>minority</td>
<td>4.35%</td>
<td>937</td>
<td>49C</td>
<td>Environment</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>UCI</td>
<td>M-class&gt;0</td>
<td>5.00%</td>
<td>1389</td>
<td>10N</td>
<td>Nature</td>
</tr>
<tr>
<td>us_crime</td>
<td>UCI</td>
<td>freq&gt;0.65</td>
<td>7.69%</td>
<td>1994</td>
<td>122C</td>
<td>Social</td>
</tr>
<tr>
<td>wine_quality</td>
<td>UCI</td>
<td>score&lt;=4</td>
<td>3.70%</td>
<td>4898</td>
<td>11C</td>
<td>Business</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>UCI</td>
<td>ME2</td>
<td>3.44%</td>
<td>1484</td>
<td>8C</td>
<td>Life</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>LIBSVM</td>
<td>8</td>
<td>7.14%</td>
<td>2417</td>
<td>103C</td>
<td>Life</td>
</tr>
</tbody>
</table>

4.2.1 Experimental Methodology

Logistic regression models, trained by the standard log-likelihood function in Section 2.2.1, the penalized log-likelihood functions in Section 2.2.2, and the proposed penalized log-likelihood function in Section 2.2.1, were compared comprehensively on each dataset, as listed below. Their performance was evaluated by 100 runs of 10-fold stratified cross validation. For each iteration in each run of the cross validation, the Area under ROC Curve (i.e. AUROC or AUC) on the validation data was computed.

1. Standard: Logistic regression model trained by the loss function in Eq. 2.16 with no penalty weights. To fit the model, the logistic regression model function from
the Scikit-Learn python package \cite{14} \cite{83} was used because its optimizer provided the
global optimal solution to Eq. 2.16.

2. Balanced: Logistic regression model trained by the loss function in Eq. 2.20 with the
balanced global penalty weights by assuming $\tau$ as 0.5 in Eq. 2.21 to adjust weights
inversely proportional to class frequencies \cite{2}. To fit the model, the logistic regression
model function with the hyperparameter “class_weight” set as “balanced” from the
Scikit-Learn python package was used because its optimizer provided the global optimal
solution to Eq. 2.20.

3. Weighted: Logistic regression model trained by the loss function in Eq. 2.20 with the
global penalty weights (i.e. $W_0$, $W_1$) in Eq. 2.21 tuned based on $\tau$ from 0 to 0.5 with
the step size 0.01. Take the dataset abalone_19 for example. Figure 4.1 shows how $W_1$,$W_0$ and AUROC change as $\tau$ increases. The $\tau$ value that generated the largest 10-fold
cross-validation mean AUROC was chosen. To fit the model, the logistic regression
model function with the hyperparameter “class_weight” from the Scikit-Learn python
package was used because its optimizer provided the global optimal solution to Eq.
2.20.

4. Kernel: Logistic regression model trained by the loss function in Eq. 2.23 with the
local kernel penalty weights in Eq. 2.24. This model was implemented by a custom
built function. $K_w$ was tuned as a hyperparameter from 0 to 1 with the step size 0.1,
capturing the nonlinearity of the model \cite{21}.

5. Learnable: Logistic regression model trained by the loss function in Eq. 3.1 with the
learnable local penalty weights $\lambda_i$ as decision variables. $\lambda_i$ were initialized to be 1
and the learning rates (i.e. $\alpha_1$, $\alpha_2$) were tuned for each dataset. This model was
implemented by a custom built function. The learning process was terminated when
the AUROC on the validation data ceased to increase for the purpose of preventing
the overfitting \cite{63}.

In the experiment of comparing the computation time of these five models, all models
were implemented with the same data structures used in the custom built function of the
learnable model, to eliminate the effect caused by the different data structures used in the Scikit-Learn python package and custom built functions.

### 4.2.2 Experimental Results

Models on each dataset are compared based on the statistics of AUROCs of 100 runs of 10-fold stratified cross validation, including 95% confidence interval, mean, and standard deviation, as well as the training time, as shown in Table 4.2. We have the following insights:

1. On 9 of all 10 datasets, the learnable models produce a higher 95% confidence interval, higher mean, and smaller standard deviation of AUROCs, as highlighted in bold in Table 4.2, compared with the standard models, balanced models, and weighted models.

2. Only on the dataset wine_quality, the kernel model generates a higher 95% confidence interval and higher mean of AUROCs than other models. The reason is that the nonlinear relationship in the dataset wine_quality is captured by the kernel model quite well. The kernel model is a nonlinear model, where $K_w$ is restricted to small values to ensure the model nonlinearity in the experiment setting, while other models are linear models. If the relationships between independent variables and the dependent variable are nonlinear, the kernel model captures their patterns better. Take the dataset wine_quality as an example. Based on the empirical logit plots in Figure 4.2, most independent variables have nonlinear relationship with the dependent variable. This leads the kernel model to perform the best.
<table>
<thead>
<tr>
<th>dataset</th>
<th>model</th>
<th>95% confidence interval</th>
<th>mean</th>
<th>std</th>
<th>training time per run(seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>abalone_19</td>
<td>standard</td>
<td>(0.8168, 0.8279)</td>
<td>0.8224</td>
<td>0.0889</td>
<td>39.1791</td>
</tr>
<tr>
<td>abalone_19</td>
<td>balanced</td>
<td>(0.8396, 0.8490)</td>
<td>0.8443</td>
<td>0.0764</td>
<td>45.4924</td>
</tr>
<tr>
<td>abalone_19</td>
<td>weighted</td>
<td>(0.8396, 0.8490)</td>
<td>0.8443</td>
<td>0.0764</td>
<td>39.3664</td>
</tr>
<tr>
<td>abalone_19</td>
<td>kernel</td>
<td>(0.7018, 0.7139)</td>
<td>0.7078</td>
<td>0.0969</td>
<td>1435.3485</td>
</tr>
<tr>
<td>abalone_19</td>
<td>learnable</td>
<td>(0.8595, 0.8681)</td>
<td>0.8638</td>
<td>0.0690</td>
<td>30.5765</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>standard</td>
<td>(0.8452, 0.8568)</td>
<td>0.8510</td>
<td>0.0935</td>
<td>19.8571</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>balanced</td>
<td>(0.8582, 0.8695)</td>
<td>0.8639</td>
<td>0.0908</td>
<td>22.4839</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>weighted</td>
<td>(0.8582, 0.8695)</td>
<td>0.8639</td>
<td>0.0908</td>
<td>22.4354</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>kernel</td>
<td>(0.5553, 0.5648)</td>
<td>0.5600</td>
<td>0.0771</td>
<td>141.7217</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>learnable</td>
<td>(0.8765, 0.8867)</td>
<td>0.8816</td>
<td>0.0823</td>
<td>15.3639</td>
</tr>
<tr>
<td>ecoli</td>
<td>standard</td>
<td>(0.9161, 0.9272)</td>
<td>0.9216</td>
<td>0.0895</td>
<td>12.1102</td>
</tr>
<tr>
<td>ecoli</td>
<td>balanced</td>
<td>(0.9077, 0.9196)</td>
<td>0.9136</td>
<td>0.0957</td>
<td>14.6242</td>
</tr>
<tr>
<td>ecoli</td>
<td>weighted</td>
<td>(0.9196, 0.9309)</td>
<td>0.9249</td>
<td>0.0851</td>
<td>14.0671</td>
</tr>
<tr>
<td>ecoli</td>
<td>kernel</td>
<td>(0.9405, 0.9548)</td>
<td>0.9431</td>
<td>0.0428</td>
<td>27.4736</td>
</tr>
<tr>
<td>ecoli</td>
<td>learnable</td>
<td>(0.9404, 0.9473)</td>
<td>0.9439</td>
<td>0.0553</td>
<td>7.3705</td>
</tr>
<tr>
<td>oil</td>
<td>standard</td>
<td>(0.9329, 0.9396)</td>
<td>0.9362</td>
<td>0.0544</td>
<td>19.1043</td>
</tr>
<tr>
<td>oil</td>
<td>balanced</td>
<td>(0.9167, 0.9263)</td>
<td>0.9215</td>
<td>0.0771</td>
<td>21.9376</td>
</tr>
<tr>
<td>oil</td>
<td>weighted</td>
<td>(0.9167, 0.9263)</td>
<td>0.9215</td>
<td>0.0771</td>
<td>21.6915</td>
</tr>
<tr>
<td>oil</td>
<td>kernel</td>
<td>(0.8832, 0.8922)</td>
<td>0.8877</td>
<td>0.0718</td>
<td>10.3713</td>
</tr>
<tr>
<td>oil</td>
<td>learnable</td>
<td>(0.9472, 0.9518)</td>
<td>0.9495</td>
<td>0.0377</td>
<td>13.9312</td>
</tr>
<tr>
<td>ozone_level</td>
<td>standard</td>
<td>(0.8936, 0.9007)</td>
<td>0.8971</td>
<td>0.0573</td>
<td>34.8695</td>
</tr>
<tr>
<td>ozone_level</td>
<td>balanced</td>
<td>(0.8725, 0.8813)</td>
<td>0.8769</td>
<td>0.0708</td>
<td>38.9631</td>
</tr>
<tr>
<td>ozone_level</td>
<td>weighted</td>
<td>(0.9006, 0.9069)</td>
<td>0.9038</td>
<td>0.0509</td>
<td>37.2606</td>
</tr>
<tr>
<td>ozone_level</td>
<td>kernel</td>
<td>(0.4838, 0.4967)</td>
<td>0.4903</td>
<td>0.1041</td>
<td>1635.0841</td>
</tr>
<tr>
<td>ozone_level</td>
<td>learnable</td>
<td>(0.9162, 0.9221)</td>
<td>0.9191</td>
<td>0.0474</td>
<td>23.1777</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>standard</td>
<td>(0.7701, 0.7818)</td>
<td>0.7759</td>
<td>0.0948</td>
<td>21.5702</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>balanced</td>
<td>(0.7610, 0.7732)</td>
<td>0.7671</td>
<td>0.0983</td>
<td>24.9768</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>weighted</td>
<td>(0.7669, 0.7790)</td>
<td>0.7730</td>
<td>0.0970</td>
<td>24.1436</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>kernel</td>
<td>(0.7341, 0.7447)</td>
<td>0.7394</td>
<td>0.0852</td>
<td>293.0257</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>learnable</td>
<td>(0.8191, 0.8282)</td>
<td>0.8236</td>
<td>0.0731</td>
<td>11.9129</td>
</tr>
<tr>
<td>us_crime</td>
<td>standard</td>
<td>(0.9173, 0.9217)</td>
<td>0.9195</td>
<td>0.0351</td>
<td>41.2519</td>
</tr>
<tr>
<td>us_crime</td>
<td>balanced</td>
<td>(0.9085, 0.9130)</td>
<td>0.9107</td>
<td>0.0363</td>
<td>51.1435</td>
</tr>
<tr>
<td>us_crime</td>
<td>weighted</td>
<td>(0.9159, 0.9202)</td>
<td>0.9180</td>
<td>0.0351</td>
<td>37.8568</td>
</tr>
<tr>
<td>us_crime</td>
<td>kernel</td>
<td>(0.7410, 0.7485)</td>
<td>0.7447</td>
<td>0.0608</td>
<td>966.8329</td>
</tr>
<tr>
<td>us_crime</td>
<td>learnable</td>
<td>(0.9290, 0.9328)</td>
<td>0.9309</td>
<td>0.0306</td>
<td>16.5772</td>
</tr>
<tr>
<td>wine_quality</td>
<td>standard</td>
<td>(0.7792, 0.7866)</td>
<td>0.7829</td>
<td>0.0600</td>
<td>52.7785</td>
</tr>
<tr>
<td>wine_quality</td>
<td>balanced</td>
<td>(0.7797, 0.7870)</td>
<td>0.7834</td>
<td>0.0588</td>
<td>57.7737</td>
</tr>
<tr>
<td>wine_quality</td>
<td>weighted</td>
<td>(0.7807, 0.7880)</td>
<td>0.7843</td>
<td>0.0593</td>
<td>44.4346</td>
</tr>
<tr>
<td>wine_quality</td>
<td>kernel</td>
<td>(0.8592, 0.8641)</td>
<td>0.8617</td>
<td>0.0395</td>
<td>203.2618</td>
</tr>
<tr>
<td>wine_quality</td>
<td>learnable</td>
<td>(0.7861, 0.7933)</td>
<td>0.7897</td>
<td>0.0581</td>
<td>54.7964</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>standard</td>
<td>(0.8679, 0.8792)</td>
<td>0.8736</td>
<td>0.0909</td>
<td>24.8167</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>balanced</td>
<td>(0.8687, 0.8793)</td>
<td>0.8740</td>
<td>0.0851</td>
<td>27.2143</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>weighted</td>
<td>(0.8704, 0.8816)</td>
<td>0.8760</td>
<td>0.0899</td>
<td>23.5347</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>kernel</td>
<td>(0.8804, 0.8915)</td>
<td>0.8859</td>
<td>0.0891</td>
<td>259.5875</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>learnable</td>
<td>(0.8937, 0.9022)</td>
<td>0.8979</td>
<td>0.0684</td>
<td>19.5705</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>standard</td>
<td>(0.5648, 0.5728)</td>
<td>0.5688</td>
<td>0.0649</td>
<td>58.8315</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>balanced</td>
<td>(0.5548, 0.5630)</td>
<td>0.5589</td>
<td>0.0657</td>
<td>64.3612</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>weighted</td>
<td>(0.5572, 0.5653)</td>
<td>0.5613</td>
<td>0.0656</td>
<td>45.2029</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>kernel</td>
<td>(0.5467, 0.5483)</td>
<td>0.5475</td>
<td>0.0123</td>
<td>167.4668</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>learnable</td>
<td>(0.6279, 0.6350)</td>
<td>0.6315</td>
<td>0.0571</td>
<td>63.2463</td>
</tr>
</tbody>
</table>
3. On the datasets abalone_19, arrhythmia and oil, the balanced models are identical to the weighted models with $\tau = 0.5$.

4. On the datasets abalone_19, arrhythmia and ozone_level, the weighted models have a higher 95% confidence interval and higher mean of AUROCs than the standard models. On the datasets oil, solar_flare_m0 and yeast_ml8, the standard models have a higher 95% confidence interval and higher mean of AUROCs than the weighted models. On the datasets ecoli, us_crime, wine_quality and yeast_me2, the standard models and weighted models are similar regarding the 95% confidence interval and mean of AUROCs.

5. On all datasets, the kernel models take the largest training time. On the datasets abalone_19, arrhythmia, ecoli, oil, ozone_level, solar_flare_m0, us_crime and yeast_me2, the learnable models take the least training time. For the dataset wine_quality and yeast_ml8, the training time of the learnable model is smaller than the balanced model, although it is slightly greater than the standard model and the weighted model.

In summary, compared with the balanced and weighted models, the learnable models perform better based on AUROCs as well as the training time in most cases. Compared with the standard models, the learnable models perform better based on AUROCs, but their training time performance depends on the datasets. Compared with the kernel models, the learnable models perform better based on training time in all cases as well as AUROCs in linear cases. Experimental results are consistent with characteristics of penalized log-likelihood functions summarized in Table 3.1.
Figure 4.2: Empirical Logit Plots of the Dataset Wine Qualität
4.3 Estimated Probability Distribution and Performance Analysis of Models

To further analyze the models in respects to the estimated probability distribution and additional performance measurements including Type I Error, Type II Error and accuracy, a more detailed study on the dataset GiveMeSomeCredit from Kaggle is conducted. It provides 10 variables representing 150,000 clients’ biographical and financial information, including Monthly Income, Debt Ratio, Age, Number of Dependents, Number of Open Credit Lines and Loans, Number of Real Estate Loans or Lines, Revolving Utilization of Unsecured Lines, Number of Time 30-59 Days Past Due Not Worse, Number of Time 60-89 Days Past Due Not Worse and Number of Times 90 Days Late [53], as shown in Table 4.3. The goal is to predict whether a client will experience financial distress in the next two years or not, indicated by the dependent variable SeriousDlqin2yrs. As shown in Table 4.4, there are 10,026 delinquency observations and 139,937 non-delinquency observations. The proportion of delinquency observations is 6.68%. 
Table 4.3: Variables for Analysis and Modeling.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeriousDlqin2yrs</td>
<td>Binary</td>
<td>Person experienced 90 days past due delinquency or worse</td>
</tr>
<tr>
<td>MonthlyIncome</td>
<td>Interval</td>
<td>Monthly income</td>
</tr>
<tr>
<td>DebtRatio</td>
<td>Interval</td>
<td>Monthly debt payments, alimony, living costs divided by monthly gross income</td>
</tr>
<tr>
<td>Age</td>
<td>Interval</td>
<td>Age of borrower in years</td>
</tr>
<tr>
<td>NumberOfDependents</td>
<td>Interval</td>
<td>Number of dependents in family excluding themselves (spouse, children, etc.)</td>
</tr>
<tr>
<td>NumberOfOpenCreditLinesAndLoans</td>
<td>Interval</td>
<td>Number of open loans (installment like car loan or mortgage) and lines of credit (e.g. credit cards)</td>
</tr>
<tr>
<td>NumberRealEstateLoansOrLines</td>
<td>Interval</td>
<td>Number of mortgage and real estate loans including home equity lines of credit</td>
</tr>
<tr>
<td>RevolvingUtilizationOfUnsecuredLines</td>
<td>Interval</td>
<td>Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits</td>
</tr>
<tr>
<td>NumberOfTime30 – 59DaysPastDueNotWorse</td>
<td>Interval</td>
<td>Number of times borrower has been 30-59 days past due but no worse in the last 2 years</td>
</tr>
<tr>
<td>NumberOfTime60 – 89DaysPastDueNotWorse</td>
<td>Interval</td>
<td>Number of times borrower has been 60-89 days past due but no worse in the last 2 years</td>
</tr>
<tr>
<td>NumberOfTimes90DaysLate</td>
<td>Interval</td>
<td>Number of times borrower has been 90 days or more past due</td>
</tr>
</tbody>
</table>
Prior to the modeling, the following three data processing steps are performed. Because the focus of this paper is the comparison of algorithms, no variable transformations are performed.

1. Drop observations with missing values because the missing data does not present any pattern based on Missing Completely at Random analysis. There are 29,731 observations with missing values in original variables provided, which are 19.82% of the total.

2. Split the dataset into 80% as training data, 10% as validation data, and 10% as test data.

3. Select important independent variables based on information values. By following the rule suggested by Hand et. al [41], the variables with the information value over 0.1 in Table 4.5 are used in the model.
### Table 4.5: Information Values.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Information Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RevolvingUtilizationOfUnsecuredLines</td>
<td>1.1635</td>
</tr>
<tr>
<td>NumberOfTime30 − 59DaysPastDueNotWorse</td>
<td>0.4865</td>
</tr>
<tr>
<td>NumberOfTimes90DaysLate</td>
<td>0.4842</td>
</tr>
<tr>
<td>NumberOfTime60 − 89DaysPastDueNotWorse</td>
<td>0.2648</td>
</tr>
<tr>
<td>Age</td>
<td>0.2620</td>
</tr>
<tr>
<td>NumberOfOpenCreditLinesAndLoans</td>
<td>0.0852</td>
</tr>
<tr>
<td>MonthlyIncome</td>
<td>0.0813</td>
</tr>
<tr>
<td>DebtRatio</td>
<td>0.0795</td>
</tr>
<tr>
<td>NumberOfDependents</td>
<td>0.0279</td>
</tr>
<tr>
<td>NumberRealEstateLoansOrLines</td>
<td>0.0184</td>
</tr>
</tbody>
</table>

#### 4.3.1 Estimated Probability Distribution

Four models are built, including standard, balanced, weighted, and learnable, described in Section 4.2.1. The weighted model is not compared in the following text because the weighted and the balanced model are the same on this dataset. The estimated probability distributions of the models on the test data can be found in Figure 4.3a, 4.3c and 4.3e, respectively, where the probabilities estimated for true event observations (i.e. 1) are colored in orange and the ones for true non-event observations (i.e. 0) are colored in blue. We have the following insights:

1. For the standard model, the estimated probabilities for most non-event observations fall in the range $[0, 0.2]$, which is the same as the estimated probabilities for most event observations. They overlap together.

2. For the balanced model, the estimated probabilities for most non-event observations fall in the range $[0.1, 0.9]$, while the estimated probabilities for most event observations fall in the range $[0.2, 1]$, which shift towards 1 a little bit compared with the non-event observations.
3. For the learnable model, the estimated probabilities for most non-event observations fall in the range $[0.9, 0.97]$, while the estimated probabilities for most event observations fall in the range $[0.95, 1]$, which shift towards 1 much more compared with the non-event observations.

It is reasonable and expected that the estimated probabilities by the learnable model are shifted towards 1 overall compared with the standard model and balanced model, because more penalty weights are applied on the misclassifications of event observations during the training phase. This can be effectively addressed by choosing an appropriate probability cutoff to achieve the best performance, which will be illustrated in Section 4.3.2. The results suggest that the learnable model differentiates the true events and true non-events better, compared with the standard model and the balanced model.

![Graphs showing predicted probabilities and probability cutoffs](image)

**Figure 4.3:** Predicted Probabilities and Probability Cutoff on Test Data
4.3.2 Performance Measurements under Probability Cutoff

The probability cutoff is set to be the probability at the intersection point of the sensitivity plot and the specificity plot in order to transform a probability to a binary decision [38] [85], for the purpose of balancing Type I Error and Type II Error. The probability cutoff for three models (i.e. standard, balanced, learnable) on the test data is 0.0675, 0.4516 and 0.9749, as shown in Figure 4.3b, 4.3d and 4.3f, respectively. Under corresponding probability cutoffs, the additional performance measurements are computed, including Type I Error, Type II Error and Accuracy, as well as AUROC, which can be found in Table 4.6. We have the following insights:

1. Compared with the standard model, the learnable model decreases Type I error by 9.01%, decreases Type II error by 8.89%, increases accuracy by 9.01%, and increases AUROC by 0.0899 on the test data.

2. Compared with the balanced model, the learnable model decreases Type I error by 1.78%, decreases Type II error by 1.80%, increases accuracy by 1.79%, and increases AUROC by 0.0179 on the test data.

3. The learnable model performs better consistently on the validation data and test data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Type I Error</th>
<th>Type II Error</th>
<th>Accuracy</th>
<th>AUROC</th>
<th>Probability Cutoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>validation data</td>
<td>standard</td>
<td>36.00%</td>
<td>36.03%</td>
<td>64.00%</td>
<td>0.6398</td>
<td>0.0672</td>
</tr>
<tr>
<td></td>
<td>balanced</td>
<td>27.14%</td>
<td>27.13%</td>
<td>72.86%</td>
<td>0.7287</td>
<td>0.4572</td>
</tr>
<tr>
<td></td>
<td>learnable</td>
<td>26.12%</td>
<td>25.93%</td>
<td>73.89%</td>
<td>0.7397</td>
<td>0.9751</td>
</tr>
<tr>
<td>test data</td>
<td>standard</td>
<td>35.75%</td>
<td>35.65%</td>
<td>64.26%</td>
<td>0.6430</td>
<td>0.0675</td>
</tr>
<tr>
<td></td>
<td>balanced</td>
<td>28.52%</td>
<td>28.47%</td>
<td>71.48%</td>
<td>0.7150</td>
<td>0.4516</td>
</tr>
<tr>
<td></td>
<td>learnable</td>
<td>26.74%</td>
<td>26.67%</td>
<td>73.27%</td>
<td>0.7329</td>
<td>0.9749</td>
</tr>
</tbody>
</table>

The learnable model performs the best according to all these performance measurements on both the validation data and the test data, followed by the balanced model and the standard model. Moreover, there is no overfitting.
4.3.3 Estimated Model Coefficients

The estimated coefficients of three models (i.e. standard, balanced, learnable) in Section 4.3.2 are examined. As shown in Table 4.7, the models have different estimated values for each independent variable, as well as the sign of the variable $NumberOfTime^{60-89}DaysPastDueNotWorse$, which is negative in the standard model and positive in the balanced model and the learnable model. Its empirical logit plot in Figure 4.4 shows a positive relationship. Based on its variance inflation factor (VIF) in Table 4.8, the multicollinearity existis among the variables $NumberOfTime^{30-59}DaysPastDueNotWorse$ and $NumberOfTimes^{90}DaysLate$, which causes the sign change. Because all of their information values are above 0.1 in Table 4.5, none of them should be dropped. Despite the multicollinearity, the balanced model and the learnable model generate the positive estimate that is consistent with the univariate effect.

Table 4.7: Estimated Model Coefficients.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Standard Model</th>
<th>Balanced Model</th>
<th>Learnable Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>intercept</td>
<td>-2.77</td>
<td>-0.19</td>
<td>3.62</td>
</tr>
<tr>
<td>age</td>
<td>-0.03</td>
<td>-0.02</td>
<td>-0.02</td>
</tr>
<tr>
<td>RevolvingUtilizationOfUnsecuredLines</td>
<td>-0.40</td>
<td>-0.40</td>
<td>-0.22</td>
</tr>
<tr>
<td>$NumberOfTime^{30-59}DaysPastDueNotWorse$</td>
<td>1.77</td>
<td>2.05</td>
<td>1.34</td>
</tr>
<tr>
<td>$NumberOfTimes^{90}DaysLate$</td>
<td>1.58</td>
<td>2.30</td>
<td>1.86</td>
</tr>
<tr>
<td>$NumberOfTime^{60-89}DaysPastDueNotWorse$</td>
<td>-3.19</td>
<td>0.87</td>
<td>1.19</td>
</tr>
</tbody>
</table>
Figure 4.4: Empirical Logit Plot of \(NumberOfTime60 - 89DaysPastDueNotWorse\)

Table 4.8: VIF for \(NumberOfTime60 - 89DaysPastDueNotWorse\)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>VIF Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>RevolvingUtilizationOfUnsecuredLines</td>
<td>1</td>
</tr>
<tr>
<td>NumberOfTime30 - 59DaysPastDueNotWorse</td>
<td>20.5</td>
</tr>
<tr>
<td>NumberOfTimes90DaysLate</td>
<td>20.5</td>
</tr>
<tr>
<td>Age</td>
<td>1</td>
</tr>
</tbody>
</table>

4.3.4 Discussions

One potential issue of this method is overfitting because local penalty weights are learned from the training data along with the model coefficients. This is a common issue for cost-sensitive learning methods [63]. The strategy we have adopted for preventing overfitting is to stop the training process when the AUROC on the validation data ceases to increase. As shown by the performance measurements on the validation data and test data of the credit dataset in Table 4.6, the generalization ability of the logistic regression model trained by the proposed log-likelihood function is good.

On the other hand, the proposed model tends to produce a higher estimated probabilities overall but shifts the estimated probability for the true class 1 much more than the true class
0, as shown in Figure 4.3. The distributions of the estimated probabilities on the 10 public datasets listed in Table 4.1 are further examined and verified by the following steps.

1. Divide the dataset into the training data, validation data, and test data by 80%, 10%, and 10%, respectively.

2. Train the model on the training data and validate the model on the validation data.

3. Estimate the probabilities of the test data by the trained model above.

4. Rank the estimated probabilities of observations in the true class 0 and 1, respectively.

5. Divide the ranked estimated probabilities into 10 bins with equal distances for the true class 0 and 1, respectively.

6. Identify the bins of the estimated probabilities with the most observations, as shown in columns “Most Frequent Bins for True Class 1” and “Most Frequent Bins for True Class 0” in Table 4.9, which correspond to the peak of the estimated probability distribution of the true class 1 highlighted in orange and the peak of the estimated probability distribution of the true class 0 highlighted in blue in Figure 4.3, respectively. If the top two bins have a similar number of observations, both of them are reported, as shown in the subcolumns “1st Bin” and “2nd Bin” in Table 4.9.
Table 4.9: The Estimated Probability Bins with the Most Observations.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>(Most Frequent Bins for True Class 1)</th>
<th>(Most Frequent Bins for True Class 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1st Bin</td>
<td>2nd Bin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1st Bin</td>
<td>2nd Bin</td>
</tr>
<tr>
<td>abalone_19</td>
<td>standard</td>
<td>(0.014, 0.016]</td>
<td>(0.011, 0.012]</td>
</tr>
<tr>
<td>abalone_19</td>
<td>balanced</td>
<td>(0.075, 0.761]</td>
<td>(0.371, 0.427]</td>
</tr>
<tr>
<td>abalone_19</td>
<td>learnable</td>
<td>(0.921, 0.965]</td>
<td>(0.742, 0.787]</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>standard</td>
<td>(0.493, 0.541]</td>
<td>(0.158, 0.206]</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>balanced</td>
<td>(0.810, 0.885]</td>
<td>(0.434, 0.509]</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>learnable</td>
<td>(0.791, 0.862]</td>
<td>(0.370, 0.440]</td>
</tr>
<tr>
<td>ecoli</td>
<td>standard</td>
<td>(0.732, 0.813]</td>
<td>(0.570, 0.651]</td>
</tr>
<tr>
<td>ecoli</td>
<td>balanced</td>
<td>(0.887, 0.985]</td>
<td>(0.789, 0.887]</td>
</tr>
<tr>
<td>ecoli</td>
<td>learnable</td>
<td>(0.875, 0.976]</td>
<td>(0.683, 0.781]</td>
</tr>
<tr>
<td>oil</td>
<td>standard</td>
<td>(0.035, 0.107]</td>
<td>(0.677, 0.749]</td>
</tr>
<tr>
<td>oil</td>
<td>balanced</td>
<td>(0.921, 0.988]</td>
<td>(0.652, 0.719]</td>
</tr>
<tr>
<td>oil</td>
<td>learnable</td>
<td>(0.875, 0.948]</td>
<td>(0.508, 0.582]</td>
</tr>
<tr>
<td>ozone_level</td>
<td>standard</td>
<td>(0.048, 0.073]</td>
<td>(0.000, 0.024]</td>
</tr>
<tr>
<td>ozone_level</td>
<td>balanced</td>
<td>(0.899, 0.999]</td>
<td>(0.000, 0.099]</td>
</tr>
<tr>
<td>ozone_level</td>
<td>learnable</td>
<td>(0.185, 0.203]</td>
<td>(0.203, 0.221]</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>standard</td>
<td>(0.002, 0.044]</td>
<td>(0.375, 0.416]</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>balanced</td>
<td>(0.829, 0.921]</td>
<td>(0.737, 0.829]</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>learnable</td>
<td>(0.500, 0.501]</td>
<td>(0.499, 0.500]</td>
</tr>
<tr>
<td>us_crime</td>
<td>standard</td>
<td>(0.008, 0.107]</td>
<td>(0.205, 0.303]</td>
</tr>
<tr>
<td>us_crime</td>
<td>balanced</td>
<td>(0.905, 0.999]</td>
<td>(0.060, 0.155]</td>
</tr>
<tr>
<td>us_crime</td>
<td>learnable</td>
<td>(0.526, 0.600]</td>
<td>(0.896, 0.970]</td>
</tr>
<tr>
<td>wine_quality</td>
<td>standard</td>
<td>(0.006, 0.018]</td>
<td>(0.018, 0.030]</td>
</tr>
<tr>
<td>wine_quality</td>
<td>balanced</td>
<td>(0.615, 0.697]</td>
<td>(0.533, 0.615]</td>
</tr>
<tr>
<td>wine_quality</td>
<td>learnable</td>
<td>(0.667, 0.725]</td>
<td>(0.609, 0.667]</td>
</tr>
<tr>
<td>yeast_mle2</td>
<td>standard</td>
<td>(0.000, 0.074]</td>
<td>(0.664, 0.737]</td>
</tr>
<tr>
<td>yeast_mle2</td>
<td>balanced</td>
<td>(0.895, 0.994]</td>
<td>(0.299, 0.398]</td>
</tr>
<tr>
<td>yeast_mle2</td>
<td>learnable</td>
<td>(0.892, 0.990]</td>
<td>(0.696, 0.794]</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>standard</td>
<td>(0.017, 0.049]</td>
<td>(0.049, 0.080]</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>balanced</td>
<td>(0.127, 0.196]</td>
<td>(0.746, 0.815]</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>learnable</td>
<td>(0.246, 0.258)</td>
<td>(0.222, 0.234)</td>
</tr>
</tbody>
</table>

Compared with the standard model, the balanced model and learnable model produce a higher estimated probabilities on all 10 datasets. Compared with the balanced model, the learnable model generates higher estimated probabilities on the three datasets (i.e. abalone_19, wine_quality, yeast_ml8). This is reasonable considering how the penalty weights
are defined in the balanced and learnable model. The penalty weights in the learnable model are decision variables and learned from the data, while the penalty weights in the balanced model are pre-defined constants based on the population and sample proportion of the class. So their patterns may vary on each dataset.

4.4 Statistical Bias and Variance Analysis

The models are further diagnosed by checking their statistical bias and variance to investigate how the proposed penalized log-likelihood function improves the model discrimination ability.

Based on the decomposition results of the squared loss in Table 4.10, we have the following insights:

1. The standard model has the smallest bias on all of these 10 datasets.

2. The standard model has the smallest variance on 5 datasets (i.e. ecoli, oil, solar_flare_m0, wine_quality, yeast_me2).

3. There is no clear pattern on the bias and variance of the weighted model, balanced model, and learnable model.

Despite the smallest bias and variance of the squared loss, it has been known that the standard model does not really differentiate the minority class and the majority class well on the imbalanced data. This indicates that the bias-variance decomposition of the squared loss may not be proper diagnosis measurement for class imbalance problems.

Based on the decomposition results of 0 – 1 loss in Table 4.11, we notice the following:

1. The learnable model has the smallest bias on 6 of these 10 datasets (i.e. ecoli, ozone_level, solar_flare_m0, us_crime, yeast_me2, yeast_ml8), as highlighted in bold.

2. The kernel model has the smallest variance on 8 of these 10 datasets (i.e. abalone_19, arrhythmia, oil, ozone_level, solar_flare_m0, us_crime, wine_quality, yeast_ml8).

3. There is no clear pattern on the bias and variance of the standard model, balanced model, and weighted model.
These results are more consistent with the AUROC results of repeated runs of 10-fold cross validation in Section 4.2.2. Hence, the bias-variance decomposition of 0-1 loss may be a more proper model diagnosis measurement for class imbalance problems.

Overall, the bias-variance decompositions of the squared loss and the 0 – 1 loss present different results. The reason is that an additional step is done to compute the bias-variance decomposition of the 0 – 1 loss by choosing a probability cutoff for determining a binary decision (0 or 1), compared with squared loss. From the squared loss to the 0 – 1 loss, the bias of the learnable model becomes smaller comparatively than other models. This indicates that the learnable model introduces the bias when estimating the event occurrence probability but reduces the bias after converting the event occurrence probability to a binary decision.

4.5 Conclusions

To improve the classification on the imbalanced data, we propose a novel penalized log-likelihood function by including penalty weights for event observations as decision variables and learning them from data along with model coefficients. Its advantages in the discrimination ability and computation efficiency are demonstrated in its application to train logistic regression models and in a comprehensive comparison study with logistic regression models trained by other penalized log-likelihood functions on 10 public datasets from multiple domains. Based on the statistics (i.e. 95% confidence interval, mean, standard deviation) of AUROCs over 100 runs of 10-fold stratified cross validation, the proposed method achieves a higher 95% confidence interval and a lower standard deviation on 9 of all 10 datasets. In a detailed study of an imbalanced credit dataset, the distributions of estimated probabilities and additional performance measurements (i.e. Type I error, Type II error, accuracy) further demonstrate better discrimination ability and good generalization ability of the proposed model. Moreover, the statistical bias and variance decomposition show that the proposed method reduces the model bias after converting the probabilities from the model outputs to binary decisions compared with other models.
Table 4.10: Bias-Variance Decomposition of Squared Loss

<table>
<thead>
<tr>
<th>dataset</th>
<th>model</th>
<th>SL bias</th>
<th>SL variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>abalone_19</td>
<td>standard</td>
<td>0.0080</td>
<td>0.0001</td>
</tr>
<tr>
<td>abalone_19</td>
<td>kernel</td>
<td>0.0410</td>
<td>0.0034</td>
</tr>
<tr>
<td>abalone_19</td>
<td>weighted</td>
<td>0.1257</td>
<td>0.0116</td>
</tr>
<tr>
<td>abalone_19</td>
<td>balanced</td>
<td>0.1273</td>
<td>0.0111</td>
</tr>
<tr>
<td>abalone_19</td>
<td>learnable</td>
<td>0.1688</td>
<td>0.0151</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>standard</td>
<td>0.0559</td>
<td>0.0079</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>weighted</td>
<td>0.0613</td>
<td>0.0164</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>balanced</td>
<td>0.0615</td>
<td>0.0164</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>learnable</td>
<td>0.1271</td>
<td>0.0501</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>kernel</td>
<td>0.2499</td>
<td>0</td>
</tr>
<tr>
<td>ecoli</td>
<td>standard</td>
<td>0.0542</td>
<td>0.0015</td>
</tr>
<tr>
<td>ecoli</td>
<td>balanced</td>
<td>0.0636</td>
<td>0.0039</td>
</tr>
<tr>
<td>ecoli</td>
<td>kernel</td>
<td>0.0641</td>
<td>0.0112</td>
</tr>
<tr>
<td>ecoli</td>
<td>weighted</td>
<td>0.0642</td>
<td>0.0011</td>
</tr>
<tr>
<td>ecoli</td>
<td>learnable</td>
<td>0.0743</td>
<td>0.0166</td>
</tr>
<tr>
<td>oil</td>
<td>standard</td>
<td>0.0342</td>
<td>0.0038</td>
</tr>
<tr>
<td>oil</td>
<td>learnable</td>
<td>0.0361</td>
<td>0.0038</td>
</tr>
<tr>
<td>oil</td>
<td>kernel</td>
<td>0.0483</td>
<td>0.0049</td>
</tr>
<tr>
<td>oil</td>
<td>weighted</td>
<td>0.0529</td>
<td>0.0100</td>
</tr>
<tr>
<td>oil</td>
<td>balanced</td>
<td>0.0529</td>
<td>0.0100</td>
</tr>
<tr>
<td>ozone_level</td>
<td>standard</td>
<td>0.0249</td>
<td>0.0031</td>
</tr>
<tr>
<td>ozone_level</td>
<td>weighted</td>
<td>0.0259</td>
<td>0.0010</td>
</tr>
<tr>
<td>ozone_level</td>
<td>balanced</td>
<td>0.0538</td>
<td>0.0183</td>
</tr>
<tr>
<td>ozone_level</td>
<td>learnable</td>
<td>0.0592</td>
<td>0.0387</td>
</tr>
<tr>
<td>ozone_level</td>
<td>kernel</td>
<td>0.118</td>
<td>0.0025</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>standard</td>
<td>0.0427</td>
<td>0.0026</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>weighted</td>
<td>0.0487</td>
<td>0.0050</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>kernel</td>
<td>0.0681</td>
<td>0.0075</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>balanced</td>
<td>0.1405</td>
<td>0.0182</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>learnable</td>
<td>0.1604</td>
<td>0.0315</td>
</tr>
<tr>
<td>us_crime</td>
<td>standard</td>
<td>0.0403</td>
<td>0.0096</td>
</tr>
<tr>
<td>us_crime</td>
<td>learnable</td>
<td>0.0479</td>
<td>0.0032</td>
</tr>
<tr>
<td>us_crime</td>
<td>weighted</td>
<td>0.049</td>
<td>0.0131</td>
</tr>
<tr>
<td>us_crime</td>
<td>balanced</td>
<td>0.0777</td>
<td>0.0189</td>
</tr>
<tr>
<td>us_crime</td>
<td>kernel</td>
<td>0.1883</td>
<td>0.0041</td>
</tr>
<tr>
<td>wine_quality</td>
<td>standard</td>
<td>0.0338</td>
<td>0.0004</td>
</tr>
<tr>
<td>wine_quality</td>
<td>kernel</td>
<td>0.0343</td>
<td>0.0043</td>
</tr>
<tr>
<td>wine_quality</td>
<td>weighted</td>
<td>0.0401</td>
<td>0.0012</td>
</tr>
<tr>
<td>wine_quality</td>
<td>learnable</td>
<td>0.0988</td>
<td>0.0357</td>
</tr>
<tr>
<td>wine_quality</td>
<td>balanced</td>
<td>0.1766</td>
<td>0.0050</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>standard</td>
<td>0.0292</td>
<td>0.0006</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>kernel</td>
<td>0.0308</td>
<td>0.0019</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>weighted</td>
<td>0.0308</td>
<td>0.0001</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>balanced</td>
<td>0.0952</td>
<td>0.0098</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>learnable</td>
<td>0.2287</td>
<td>0.0146</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>standard</td>
<td>0.0749</td>
<td>0.0104</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>learnable</td>
<td>0.1075</td>
<td>0.0318</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>weighted</td>
<td>0.1477</td>
<td>0.0373</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>balanced</td>
<td>0.1801</td>
<td>0.0451</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>kernel</td>
<td>0.2390</td>
<td>0.0005</td>
</tr>
<tr>
<td>dataset</td>
<td>model</td>
<td>CL bias</td>
<td>CL variance</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>abalone_19</td>
<td>standard</td>
<td>0.2927</td>
<td>0.1011</td>
</tr>
<tr>
<td>abalone_19</td>
<td>balanced</td>
<td>0.2943</td>
<td>0.0838</td>
</tr>
<tr>
<td>abalone_19</td>
<td>weighted</td>
<td>0.2943</td>
<td>0.0840</td>
</tr>
<tr>
<td>abalone_19</td>
<td>learnable</td>
<td>0.3046</td>
<td>0.0799</td>
</tr>
<tr>
<td>abalone_19</td>
<td>kernel</td>
<td>0.4163</td>
<td>0.0652</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>standard</td>
<td>0.2794</td>
<td>0.1499</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>balanced</td>
<td>0.2868</td>
<td>0.1524</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>weighted</td>
<td>0.2868</td>
<td>0.1521</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>learnable</td>
<td>0.3235</td>
<td>0.2110</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>kernel</td>
<td>0.9044</td>
<td>0.0058</td>
</tr>
<tr>
<td>ecoli</td>
<td>learnable</td>
<td>0.1089</td>
<td>0.0230</td>
</tr>
<tr>
<td>ecoli</td>
<td>balanced</td>
<td>0.1584</td>
<td>0.0430</td>
</tr>
<tr>
<td>ecoli</td>
<td>kernel</td>
<td>0.1584</td>
<td>0.0436</td>
</tr>
<tr>
<td>ecoli</td>
<td>weighted</td>
<td>0.1683</td>
<td>0.0507</td>
</tr>
<tr>
<td>ecoli</td>
<td>standard</td>
<td>0.1782</td>
<td>0.0512</td>
</tr>
<tr>
<td>oil</td>
<td>standard</td>
<td>0.1738</td>
<td>0.1020</td>
</tr>
<tr>
<td>oil</td>
<td>balanced</td>
<td>0.1879</td>
<td>0.1026</td>
</tr>
<tr>
<td>oil</td>
<td>weighted</td>
<td>0.1879</td>
<td>0.1028</td>
</tr>
<tr>
<td>oil</td>
<td>learnable</td>
<td>0.1950</td>
<td>0.0810</td>
</tr>
<tr>
<td>oil</td>
<td>kernel</td>
<td>0.3333</td>
<td>0.0603</td>
</tr>
<tr>
<td>ozone_level</td>
<td>learnable</td>
<td>0.1721</td>
<td>0.0882</td>
</tr>
<tr>
<td>ozone_level</td>
<td>weighted</td>
<td>0.2011</td>
<td>0.0747</td>
</tr>
<tr>
<td>ozone_level</td>
<td>balanced</td>
<td>0.2076</td>
<td>0.0884</td>
</tr>
<tr>
<td>ozone_level</td>
<td>standard</td>
<td>0.2129</td>
<td>0.0776</td>
</tr>
<tr>
<td>ozone_level</td>
<td>kernel</td>
<td>0.5782</td>
<td>0.0562</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>learnable</td>
<td>0.2734</td>
<td>0.1153</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>weighted</td>
<td>0.3213</td>
<td>0.1116</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>balanced</td>
<td>0.3261</td>
<td>0.1128</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>standard</td>
<td>0.3357</td>
<td>0.1111</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>kernel</td>
<td>0.3981</td>
<td>0.0945</td>
</tr>
<tr>
<td>us_crime</td>
<td>learnable</td>
<td>0.1419</td>
<td>0.0547</td>
</tr>
<tr>
<td>us_crime</td>
<td>standard</td>
<td>0.1503</td>
<td>0.0764</td>
</tr>
<tr>
<td>us_crime</td>
<td>weighted</td>
<td>0.1519</td>
<td>0.0750</td>
</tr>
<tr>
<td>us_crime</td>
<td>balanced</td>
<td>0.1636</td>
<td>0.0749</td>
</tr>
<tr>
<td>us_crime</td>
<td>kernel</td>
<td>0.2905</td>
<td>0.0671</td>
</tr>
<tr>
<td>wine_quality</td>
<td>kernel</td>
<td>0.2524</td>
<td>0.0772</td>
</tr>
<tr>
<td>wine_quality</td>
<td>standard</td>
<td>0.2673</td>
<td>0.0869</td>
</tr>
<tr>
<td>wine_quality</td>
<td>learnable</td>
<td>0.2741</td>
<td>0.1035</td>
</tr>
<tr>
<td>wine_quality</td>
<td>weighted</td>
<td>0.2816</td>
<td>0.0814</td>
</tr>
<tr>
<td>wine_quality</td>
<td>balanced</td>
<td>0.3000</td>
<td>0.0786</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>learnable</td>
<td>0.1300</td>
<td>0.0915</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>balanced</td>
<td>0.1659</td>
<td>0.0752</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>weighted</td>
<td>0.1794</td>
<td>0.0574</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>standard</td>
<td>0.1906</td>
<td>0.0597</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>kernel</td>
<td>0.2444</td>
<td>0.0771</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>learnable</td>
<td>0.4270</td>
<td>0.2353</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>standard</td>
<td>0.4793</td>
<td>0.2248</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>weighted</td>
<td>0.4821</td>
<td>0.2232</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>balanced</td>
<td>0.4876</td>
<td>0.2255</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>kernel</td>
<td>0.8320</td>
<td>0.0118</td>
</tr>
</tbody>
</table>
Chapter 5

Improving Neural Network

5.1 Introduction

Neural network has become a popular method for classification and prediction ever since the backpropagation algorithm to efficiently train neural network models was proposed by Rumelhart, Hinton, and Williams [92], 14 years after it was proposed by Werbos [108] [49]. Given its model architecture, it can learn complex nonlinear relationships from data, despite low interpretability. It has been used in broad areas such as bankruptcy prediction [81], multispectral image classification [44], face detection [90], credit scoring [109], electroencephalograph signal detection [99], heart sound classification [37], and software defect prediction [6].

However, when classifying the imbalanced data, neural network models are biased towards the majority class and misclassifies the minority class severely, despite high overall accuracy [59] [5]. One example is that in a bankruptcy prediction study where the proportion of the minority class (i.e. bankruptcy) is 0.12%, the accuracy of the neural network model is 99.23%, but its Type II Error is 92.94%, indicating 92.94% of bankruptcy observations are misclassified as non-bankruptcy [115]. The biased solution can cause great practical loss if deployed in the decision-making process.

The bias is caused by the assumption of the optimization objective (i.e. log-likelihood function) used to estimate the neural network. It assumes that the target classes are equally
distributed and all types of misclassifications are equal, which essentially maximizes the overall accuracy [63] [86] [43]. These assumptions are violated on the imbalanced data.

To mitigate the bias on the imbalanced data, one commonly used strategy is to apply misclassification-type-dependent costs in the learning process, referred to as cost-sensitive neural network [63] [105]. For example, in the bankruptcy prediction problem, it is usually more costly to misclassify the minority class (i.e. bankruptcy) as the majority class (i.e. non-bankruptcy) than misclassifying the majority class as the minority class, so more costs are expected to be added to the former than the latter. Since misclassification costs are problem-dependent, many researchers have discussed how to determine these misclassification costs in their problem contexts [33] [100], making them not generically applicable.

In the present work, the proposed penalized log-likelihood objective function in Chapter 3 is used to train neural networks, with the penalty weights for the misclassification costs of the minority class learned from data along with the model weights via the gradient descent method. Based on our experimental results on 10 public imbalanced datasets, the discrimination ability of models trained by the proposed method is improved without increasing the computational complexity.

This chapter is structured as follows. Section 5.2 illustrates the experiments and results. In Section 5.3, the estimated probability distributions of the resulting models are examined, as well as an additional performance analysis. In Section 5.4, the conclusions are presented.

5.2 Experiments

5.2.1 Experimental Methodology

Neural network models, trained by the standard log-likelihood function in Section 2.2.3 and the proposed penalized log-likelihood function in Section 3.5, were compared on 10 public imbalanced datasets listed in Table 4.1. Their performance was evaluated by 20 repeated runs of 10-fold stratified cross validation and measured by the AUROC on the validation data.
1. Standard: A neural network model trained by the loss function in Eq. 2.37 with no penalty weights. The model architecture was designed to have one hidden layer with the number of hidden neurons the same as the number of independent variables.

2. Learnable: A neural network model trained by the loss function in Eq. 3.8 with the penalty weights $\lambda_i$ for event observations as decision variables. $\lambda_i$ was initialized to be 1, and its learning rate was tuned for each dataset. The model architecture was the same as the standard model. The learning process was terminated when AUROC on the validation data ceased to increase to prevent the overfitting [63].

Because the time complexity of optimization problems for training the standard model and learnable model was the same, their computation time was not compared in the experiments.

### 5.2.2 Experimental Results

For each model on each dataset, AUROCs on the validation data over 20 repeated runs of 10-fold cross validation are summarized by their 95% confidence interval, mean, and standard deviation, as shown in Table 5.1. We have the following insights.

1. On 9 of all 10 datasets except us_crime, the learnable models produce higher 95% confidence interval, higher mean, and smaller standard deviation of AUROCs, compared with the standard models, as highlighted in bold.

2. For the dataset us_crime, the learnable model and the standard model have very similar 95% confidence intervals, means, and standard deviations of AUROCs.
Table 5.1: Experimental Results of Neural Network Models.

<table>
<thead>
<tr>
<th>dataset</th>
<th>model</th>
<th>95% confidence interval</th>
<th>mean</th>
<th>std</th>
</tr>
</thead>
<tbody>
<tr>
<td>abalone_19</td>
<td>standard</td>
<td>(0.6956, 0.7381)</td>
<td>0.7168</td>
<td>0.1522</td>
</tr>
<tr>
<td>abalone_19</td>
<td>learnable</td>
<td>(0.8047, 0.8303)</td>
<td>0.8175</td>
<td>0.0912</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>standard</td>
<td>(0.7911, 0.8343)</td>
<td>0.8127</td>
<td>0.1547</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>learnable</td>
<td>(0.8944, 0.9152)</td>
<td>0.9048</td>
<td>0.0744</td>
</tr>
<tr>
<td>ecoli</td>
<td>standard</td>
<td>(0.8788, 0.9044)</td>
<td>0.8916</td>
<td>0.0920</td>
</tr>
<tr>
<td>ecoli</td>
<td>learnable</td>
<td>(0.9238, 0.9413)</td>
<td>0.9325</td>
<td>0.0627</td>
</tr>
<tr>
<td>oil</td>
<td>standard</td>
<td>(0.8912, 0.9116)</td>
<td>0.9014</td>
<td>0.0733</td>
</tr>
<tr>
<td>oil</td>
<td>learnable</td>
<td>(0.9295, 0.9429)</td>
<td>0.9362</td>
<td>0.0478</td>
</tr>
<tr>
<td>ozone_level</td>
<td>standard</td>
<td>(0.8739, 0.8897)</td>
<td>0.8818</td>
<td>0.0565</td>
</tr>
<tr>
<td>ozone_level</td>
<td>learnable</td>
<td>(0.8986, 0.9121)</td>
<td>0.9053</td>
<td>0.0483</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>standard</td>
<td>(0.7041, 0.7263)</td>
<td>0.7152</td>
<td>0.0793</td>
</tr>
<tr>
<td>solar_flare_m0</td>
<td>learnable</td>
<td>(0.7291, 0.7473)</td>
<td>0.7382</td>
<td>0.0652</td>
</tr>
<tr>
<td>us_crime</td>
<td>standard</td>
<td>(0.8977, 0.9080)</td>
<td>0.9028</td>
<td>0.0369</td>
</tr>
<tr>
<td>us_crime</td>
<td>learnable</td>
<td>(0.8975, 0.9081)</td>
<td>0.9028</td>
<td>0.0381</td>
</tr>
<tr>
<td>wine_quality</td>
<td>standard</td>
<td>(0.7341, 0.7541)</td>
<td>0.7441</td>
<td>0.0714</td>
</tr>
<tr>
<td>wine_quality</td>
<td>learnable</td>
<td>(0.7514, 0.7693)</td>
<td>0.7603</td>
<td>0.0638</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>standard</td>
<td>(0.8408, 0.8665)</td>
<td>0.8536</td>
<td>0.0918</td>
</tr>
<tr>
<td>yeast_me2</td>
<td>learnable</td>
<td>(0.9156, 0.9300)</td>
<td>0.9228</td>
<td>0.0513</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>standard</td>
<td>(0.5621, 0.5801)</td>
<td>0.5711</td>
<td>0.0645</td>
</tr>
<tr>
<td>yeast_ml8</td>
<td>learnable</td>
<td>(0.5909, 0.6086)</td>
<td>0.5997</td>
<td>0.0635</td>
</tr>
</tbody>
</table>

To show the improvement of the learnable model explicitly, ROC curves of a 10-fold cross validation run on the dataset ecoli are presented in Figure 5.1. The mean ROC curve of the learnable model in Figure 5.1b, indicated by the black line, is closer to the upper left corner than the standard model in Figure 5.1a. Moreover, the ROCs of the learnable model are closer to each other than the standard model, also reflected by the standard deviation of AUROCs (i.e. AUC).
Figure 5.1: ROC Curves of standard vs. learnable
5.3 Estimated Probability Distribution and Performance Analysis of Models

The estimated probability distributions and additional performance measurements (i.e. Type I Error, Type II Error, Accuracy) of the resulting models on the dataset ecoli are further analyzed. The problem related to this dataset is to classify whether the cellular localization site of proteins is an inner membrane uncleavable signal sequence (imU) or not, based on the information including McGeoch’s method for signal sequence recognition, von Heijne’s method for signal sequence recognition, von Heijne’s Signal Peptidase II consensus sequence score, presence of charge on N-terminus of predicted lipoproteins, score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins, score of the ALOM membrane spanning region prediction program, and score of ALOM program after excluding putative cleavable signal regions from the sequence [46]. The proportion of imU is 10.42%, as shown in Table 5.2.

There is no missing value in the dataset, so no data cleaning is performed. The dataset is directly splitted into 80% as training data, 10% as validation data, and 10% as test data.

<table>
<thead>
<tr>
<th>Table 5.2: Frequency of Dependent Variable in the Dataset Ecoli.</th>
</tr>
</thead>
<tbody>
<tr>
<td>imU</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

5.3.1 Estimated Probability Distribution

Two neural network models are built, including standard and learnable, described in Section 5.2.1. The estimated probability distributions of the resulting models on the test data can be found in Figure 5.2a and 5.2c, respectively, where the probabilities estimated for true event observations (i.e. 1) are colored in orange and the ones for true non-event observations (i.e. 0) are colored in blue. We have the following insights:
1. For the standard model, the estimated probabilities for most true non-event observations fall in the range \((0, 0.7)\), while the estimated probabilities for most true event observations spread over \((0, 1)\).

2. For the learnable model, the estimated probabilities for most true non-event observations fall in the range \((0, 0.2)\), while the estimated probabilities for most true event observations are above 0.3.

Compared with the standard model, the learnable model estimates probabilities for true events towards 1 and probabilities for true non-events towards 0 overall, indicating that the learnable model differentiates events and non-events better.

5.3.2 Performance Measurements under Probability Cutoff

Based on the estimated probabilities, binary classification decisions are made by choosing the probability cutoff as the intersection point of the sensitivity plot and the specificity plot [38] [85], for the purpose of balancing Type I Error and Type II Error. The probability cutoff for two models (i.e. standard, learnable) are 0.0492 and 0.3389 on the test data, as
shown in Figure 5.2b and 5.2d, respectively. Under these probability cutoffs, the additional performance measurements for each model are computed correspondingly, including Type I Error, Type II Error and Accuracy, as well as AUROC in Table 5.3. We have the following insights:

1. On the validation data, compared with the standard model, the learnable model reduced Type I Error by 42.85%, reduced Type II Error by 33.33%, increased Accuracy by 41.94%, and increased AUROC by 0.1528.

2. On the test data, compared with the standard model, the learnable model reduced Type I Error by 3.34%, reduced Type II Error by 25.00%, increased Accuracy by 6.41%, and increased AUROC by 0.1834.

3. The learnable model performs better consistently on the validation data and test data.

The results above conclude that the learnable model performs the best according to all performance measurements on both the validation data and the test data.

Table 5.3: Performance Measurements on Validation Data and Test Data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Type I Error</th>
<th>Type II Error</th>
<th>Accuracy</th>
<th>AUROC</th>
<th>Probability Cutoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>validation data</td>
<td>standard</td>
<td>42.85%</td>
<td>33.33%</td>
<td>58.06%</td>
<td>0.8452</td>
<td>2.79e-09</td>
</tr>
<tr>
<td></td>
<td>learnable</td>
<td>0.00%</td>
<td>0.00%</td>
<td>100.00%</td>
<td>1.0000</td>
<td>0.0235</td>
</tr>
<tr>
<td>test data</td>
<td>standard</td>
<td>6.67%</td>
<td>25.00%</td>
<td>91.17%</td>
<td>0.7833</td>
<td>0.0492</td>
</tr>
<tr>
<td></td>
<td>learnable</td>
<td>3.33%</td>
<td>0.00%</td>
<td>97.58%</td>
<td>0.9667</td>
<td>0.3389</td>
</tr>
</tbody>
</table>

5.3.3 Discussions

As the same for all cost-sensitive learning methods [63], overfitting is a potential issue of the proposed method, since there are more parameters to learn from the training data. To prevent overfitting, the training process is stopped when the AUROC on the validation data ceases to increase. As demonstrated by the performance measurements on the validation data and test data of the ecoli dataset in Table 5.3, the generalization ability of the neural network model trained by the proposed log-likelihood function is good.
5.4 Conclusions

Imbalanced data exists in various domains, such as transportation [52], higher education [76], social science [98], and business [67]. It has received much attention from the researchers in the field of statistics, data science, and machine learning. To better solve the problem and differentiate the majority class and the minority class in the imbalanced data, we propose a novel penalized log-likelihood function by including penalty weights for misclassification costs of observations in the minority class as decision variables and learning them from data along with model weights. The proposed approach is applied to train neural network models and produces the improved performance on 10 public datasets without increasing computational complexity, compared with neural network models trained by the standard log-likelihood function, according to the statistics (i.e. 95% confidence interval, mean, and standard deviation) of AUROCs on the validation data over 20 repeated runs of 10-fold stratified cross validation. Moreover, its generalization ability is demonstrated to be good by examining the estimated probability distributions and additional performance measurements (i.e. Type I Error, Type II Error, Accuracy) on both validation data and test data of the ecoli dataset. The prediction accuracy for the majority class and the minority class are impressively improved.
Chapter 6

Conclusions

The goal of this dissertation is to make contributions to addressing challenges and improving solutions for class imbalance problems. A class imbalance problem is the task of classifying the imbalanced data with an unequal target class distribution. In the imbalanced data, there are many more observations in the majority class (i.e. non-event) than the minority class (i.e. event). The imbalanced data exists in almost every area, such as fraud detection, bankruptcy prediction, credit scoring, rare disease diagnosis, heart sound classification, and software defect prediction. For example, in the bankruptcy prediction problem, there are many more observations in non-bankruptcy (majority class) than bankruptcy (minority class).

Standard statistics and machine learning models do not differentiate the majority class and the minority class effectively in a class imbalance problem. They tend to be biased towards the majority class and severely misclassify the minority class as the majority class. However, the minority class is usually the class of interest and costs more if misclassified. For example, in a bankruptcy prediction study conducted by Zhang et. al, where the proportion of bankruptcy observations (i.e. event rate) is 0.12%, most models can achieve the overall accuracy as high as 99%, but produce Type II Error higher than 87%, indicating that more than 87% of bankruptcy observations are misclassified as non-bankruptcy. This can bring great loss in practice, for example, if banks approve loans to organizations with low predicted but actual high bankruptcy probabilities. In the same study, when the event rate is increased to 50% by resampling the data, all models reduce Type II Error to 15% with overall accuracy above 70%.
The reason the standard models are biased on the imbalanced data is that their training optimization objective function is formulated under two assumptions: 1) the target classes are equally distributed; 2) the misclassifications of the minority class and the majority class are equal, which essentially makes the objective to maximize the overall accuracy. Both of these two assumptions are violated on the imbalanced data.

Many researchers have proposed solutions to mitigate such bias from the perspectives of input data (e.g., sampling), feature (e.g., feature selection, variable discretization), algorithm (e.g., cost-sensitive learning, ensemble learning, active learning, kernel-based learning, and one-class learning) and output data (e.g., thresholding). Besides the model development, in the model evaluation, researchers also point out that the classification model on the imbalanced data should be evaluated comprehensively by combining curve-based measurements (e.g., receiver operating characteristic curve, precision-recall curve) and single-value measurements (e.g., Type I Error, Type II Error, F1 Score, Accuracy), instead of purely on accuracy.

Among all existing approaches, cost-sensitive learning specifies different penalty weights for misclassification costs of the majority class and the minority class in the training optimization objective function. For logistic regression and neural network, the training optimization objective function is the log-likelihood function. To apply logistic regression on imbalanced data, misclassification costs in the log-likelihood objective function have been proposed to be penalized by either global penalty weights or local penalty weights. Global penalty weights are determined by the population proportion and the sample proportion of the minority class. However, it is hard to accurately estimate the population proportion of the minority class, which ultimately influences the model performance. Local penalty weights are determined by the Gaussian kernel, which is very computationally expensive and does not apply to big data. For neural network, misclassification costs are usually penalized by a constant misclassification-type-dependent cost matrix, which is derived based on particular problem contexts, making them not generic.

To overcome the challenges above, a novel penalized log-likelihood function is proposed to estimate models for better differentiating the minority class and the majority class in the imbalanced data. Compared with the pre-defined constant penalty weights for
misclassification costs in existing solutions, the penalty weights for misclassification costs of observations in the minority class are included as decision variables in the proposed log-likelihood function and learned from the data along with model coefficients/parameters during the training process. The optimization of the proposed objective function is a nonlinear programming problem, which is efficiently solved by the modified gradient descent algorithm.

The proposed penalized log-likelihood function is applied to train logistic regression and neural network models. It improves the model performance while maintaining (or even reducing) the computing complexity when compared with the standard and existing penalized log-likelihood functions.

The models are compared on 10 public datasets from multiple domains (e.g., business, environment, life). And they are evaluated over the repeated runs of 10-fold stratified cross validation: 100 runs for logistic regression and 20 runs for neural network. The performance measurement is Area under ROC Curve on the validation data, which is summarized by its statistics (i.e. 95% confidence interval, mean, standard deviation) over all runs of cross-validation. For logistic regression, on 9 of all 10 datasets, the proposed approach is more accurate, demonstrated by higher 95% confidence intervals and smaller or similar standard deviations. Only on one dataset (i.e. wine_quality), the kernel model produces a higher 95% confidence interval (0.8592, 0.8641) compared with the 95% confidence interval (0.7861, 0.7933) of the proposed model, but its training time (i.e. 2013.2618 seconds) is much longer than other models including the proposed one (i.e. 54.7964 seconds). For neural network, on 9 of all 10 datasets, the proposed method yields better results. Only on the dataset us_crime, the proposed model generates the 95% confidence interval (0.8975, 0.9081), which is similar to the 95% confidence interval (0.8977, 0.9080) by the standard model.

The estimated probability distributions are further examined and compared on the GiveMeSomeCredit dataset for logistic regression and the ecoli dataset for neural network in detail. Compared with other models, the estimated probabilities of the minority class by the proposed model shifts towards 1 much more than the majority class, indicating better model differentiation ability, which is also examined and verified on all 10 public datasets. Moreover, based on additional performance measurements (i.e. Type I Error, Type II Error,
Accuray) on both the validation data and the test data, the performance of the proposed model is improved significantly, indicating good model generalization ability.

Moreover, the estimated model coefficients are compared on the GiveMeSomeCredit dataset to show that the interpretability of logistic regression is maintained in the proposed method. The proposed model has different but reasonable estimated values for each independent variable and helps keep the univariate and multivariate effects consistent.

Furthermore, the models are diagnosed by decomposing the model errors into the statistical bias and variance to investigate how the proposed method improves the model performance. Results show that the bias and variance decomposition of the 0-1 loss is more proper for class imbalance problems, compared with the squared loss. The proposed approach reduces the model bias after converting the estimated probabilities to binary decisions, compared with other models.

One potential issue of the proposed approach is overfitting, considering there are more parameters to learn from the training data. This is a common issue for all cost-sensitive learning methods. To prevent overfitting, the strategy we have adopted is to stop the training process when the AUROC on the validation data ceases to increase.

In conclusion, to better differentiate the minority and majority classes in the imbalanced data, we propose a novel penalized log-likelihood function by including penalty weights as decision variables for misclassification costs of observations in the minority class and learn them from data along with model coefficients/parameters. This is applied to improve logistic regression and neural network. Experimental results demonstrate better model differentiation ability and good model generalization ability while reducing or maintaining the computational complexity, compared with the models trained by other variants of log-likelihood functions. For that, the proposed approach can be used broadly to solve class imbalance problems existing in almost every area.
Bibliography


92


tutorial. *Computer*, 29(3):31–44. 75

statistical learning*, volume 112. Springer. 39

*Proc. of the Intl Conf. on Artificial Intelligence*. 6

vehicle crash injury severity: A hybrid approach for imbalanced data. *Accident Analysis
& Prevention*, 120:250–261. 83


International Joint Conference on*, pages 1756–1761. IEEE. 30

repeated hold-out and bootstrap. *Computational statistics & data analysis*, 53(11):3735–
3745. 37

boosting algorithm with over-sampling to resolve data imbalance problem for bankruptcy

9(2):137–163. 2, 29, 54

*GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58. 8


In 1990 IJCNN International Joint Conference on neural networks, pages 163–168. IEEE.


[83] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel,
M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D.,


and Sutrisna, B. (2010). Prevalence and predictors of undiagnosed diabetes mellitus in
indonesia. Age, 46(53.6):100–100.

the AAAI2000 workshop on imbalanced data sets, pages 1–3.

[87] Qazi, N. and Raza, K. (2012). Effect of feature selection, smote and under sampling on
class imbalance classification. In 2012 UKSim 14th International Conference on Computer
Modelling and Simulation, pages 145–150. IEEE.


[89] Ramentol, E., Gondres, I., Lajes, S., Bello, R., Caballero, Y., Cornelis, C., and Herrera,
F. (2016). Fuzzy-rough imbalanced learning for the diagnosis of high voltage circuit
breaker maintenance: The smote-frst-2t algorithm. Engineering Applications of Artificial
Intelligence, 48:134–139.


Appendix
A Mathematical Preliminaries

A.1 Empirical Logit Plotting

When the dependent variable is binary and all independent variables are interval, the empirical logit plot is used to examine whether the relationship between the dependent variable and an independent variable is linear or not. If linear, we can use the interval form of that independent variable. If not linear, we might need to discretize that independent variable to represent the nonlinearity. Moreover, through the empirical logit plots, we can check the univariate effects, positive or negative.

The empirical logit plot is created in the following steps.

1. For each interval variable, generate percentile ranks from 1 to 100.

2. For each rank $i$ of each interval variable, calculate the total number of observations $N_i$, the number of delinquency observations $Y_i$, and the mean of the interval variable $\bar{x}_i$.

3. For each rank $i$ of each interval variable, compute the empirical logit using the formula $elogit_i = \log\left(\frac{Y_i + 0.5}{N_i - Y_i + 0.5}\right)$.

4. For each interval variable, plot the empirical logit $elogit$ against the mean in each rank $\bar{x}$ and their linear regression line. Each point in the plot represents $N_i$ data points from the data set by their mean.

5. For each interval variable, plot the empirical logit $elogit$ against the rank $i$ and their linear regression line. Each point in the plot represents $N_i$ data points from the data set by their rank index.