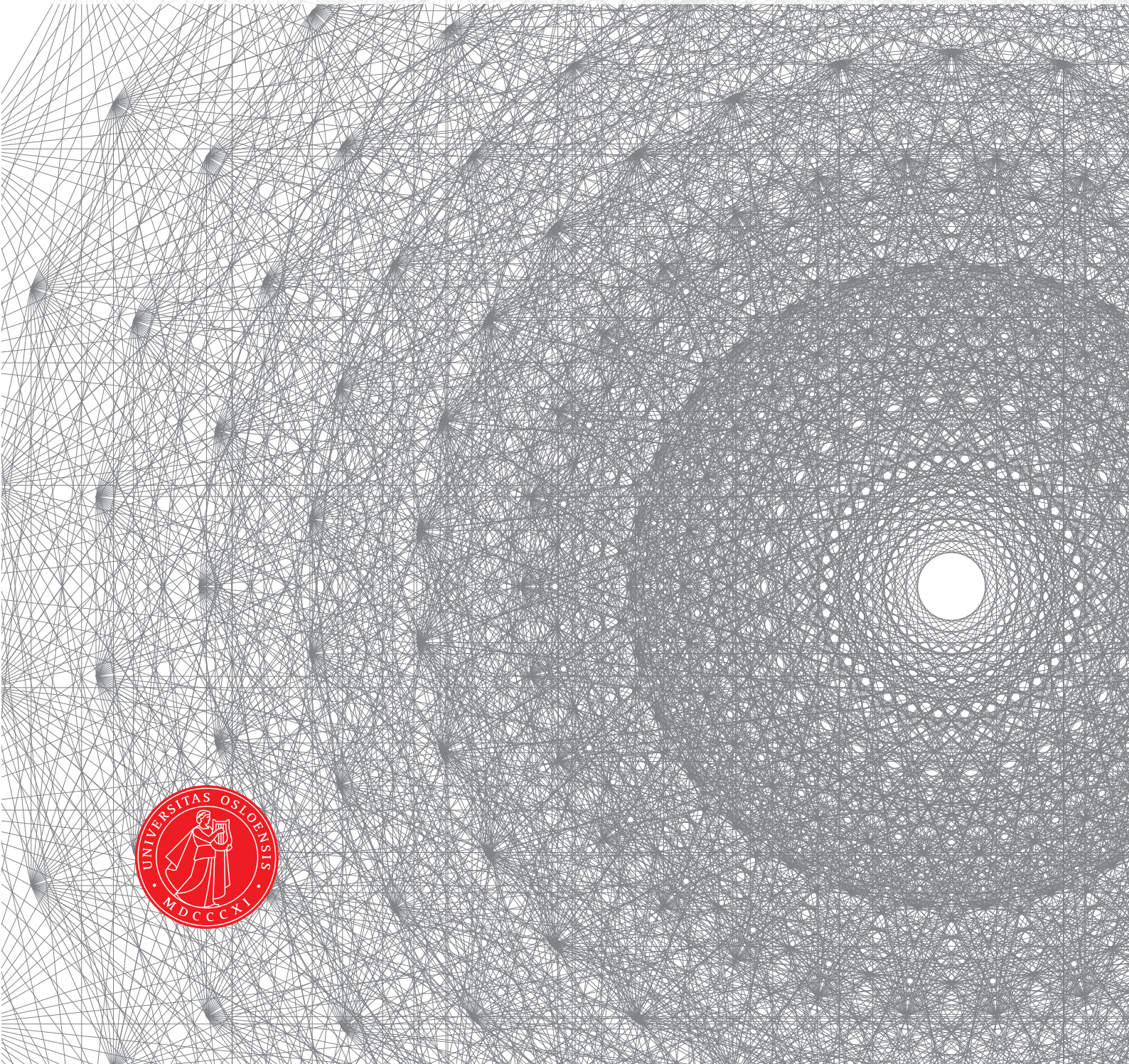
UiO : Department of Mathematics
University of Oslo

The application of penalized logistic regression for fraud detection

Studying measures of prediction performance for class imbalanced and high-dimensional data

Shuijing Liao

Master's Thesis, Autumn 2022



This master's thesis is submitted under the master's programme *Stochastic Modelling, Statistics and Risk Analysis*, with programme option *Statistics*, at the Department of Mathematics, University of Oslo. The scope of the thesis is 60 credits.

The front page depicts a section of the root system of the exceptional Lie group E_8, projected into the plane. Lie groups were invented by the Norwegian mathematician Sophus Lie (1842–1899) to express symmetries in differential equations and today they play a central role in various parts of mathematics.

Abstract

Typically, fraud data are class imbalanced, and possibly high-dimensional. They have a mixture of discrete and continuous covariates, among which a certain dependence structure exists. Building good prediction models for detecting the fraudulent cases faces challenges, for instance, due to inappropriate measures of prediction performance. We study, in the setting of fraud detection, whether and how the prediction performance of a penalized logistic regression model may be improved by applying appropriate optimality measures in cross-validation of the penalty parameters. This includes investigating the effect of different optimality measures on the prediction performance of penalized logistic regression methods, and exploring different performance measures used in the validation phase. A simulation study and an illustration on a real credit card default data, are conducted. Copulas is utilised in the simulation study, for generating data that share the aforementioned characteristics of fraud data. The results from the simulation study show that, for class imbalanced and high-dimensional data, the prediction performance of the elastic net, in terms of the AUC, the AUCPR, the AGm and the variable selection property, is significantly boosted by the optimality measure AUCPR used in cross-validation of the penalty parameters. Meanwhile, the illustration on real data shows that, the optimality measure AUCPR, only improves the AUC and the AUCPR of the lasso and elastic net regression in the third decimal. We conclude that, whether and how the optimality measure will affect the prediction performance, depends on many factors, such as the learning method, the type of data, etc. Nevertheless, we recommend reporting the AUCPR for model selection in the setting of fraud detection, especially when the AUC gives overly optimistic results.

Acknowledgements

The whole experience of this research work was memorable, and the learning process was challenging but fun. I would like to thank my supervisor Ingrid Hobæk Haff for her guidance, support, patience and encouragement throughout the the whole period of my master thesis.

Contents

Chapter 1

Introduction

Economically, fraudulent activities may cause potentially serious losses for many financial institutions, such as insurance companies, banks and the Norwegian tax administration, etc. Therefore, they tend to prioritise fraud detection highly. Fraud detection involves the task of detecting whether an actual case is fraudulent or not, which may be done by prediction models that are built with the help of regression and machine leaning techniques, for instance, logistic models.

Typically, there are a large amount of cases, and only a small proportion of them are fraudulent, resulting in class imbalanced data. Class imbalance may impair the prediction performance of many of the regression and machine learning methods used for fraud detection (Japkowicz and Stephen, 2002). Possible solutions have been proposed, for instance, re-sampling methods that manipulate the imbalance level of the training data before estimating the model (Batista et al., 2004; Chawla et al., 2002; Estabrooks et al., 2004). However, oversampling, under-sampling and SMOTE were not promising in improving the prediction performance, in terms of the AUC, of logistic models in Halsteinslid (2019).

In comparison to the development of methods for building the classifier with imbalanced data, choosing appropriate measures of prediction performance is often underestimated. In the earlier studies which dealt with class imbalanced data, the AUC was frequently used as the measure of prediction performance for model selection (Blagus and Lusa, 2010; Burez and Van den Poel, 2009). However, typical measures like the accuracy and the AUC mainly measure how well the model predicts the non-fraudulent cases, as they are in majority, whereas the main interest within fraud detection is how well the model detects fraudulent cases. The choice of measure of prediction performance needs to be made with care, for class imbalanced data within fraud detection.

The AUCPR, which measures the average precision, is recommended for highly class imbalanced data (Berrar, 2019). Additionally, the precision-recall curve has been suggested as an alternative to the ROC curve for class imbalance (Davis and Goadrich, 2006; He and Edwardo A, 2009). Meanwhile, Saito and Rehmsmeier (2015) have shown that the precision-recall plot is more informative than the ROC plot for imbalanced data, as the former emphasizes the performance on the top-ranked cases. Therefore, the corresponding summary metric AUCPR may be an appropriate performance measure for class imbalanced data within fraud detection, despite the fact it is less frequently used than the AUC. Several other measures are also included and discussed in our study, such as the adjusted geometric mean of the TPR and TNR (Batuwita and Palade, 2009), Brier score (Brier, 1950), etc.

Here, the penalized logistic regression models will be used for fraud detection, since the data one encounters in fraud detection are also possibly high-dimensional, which may introduce variance and cause overfitting. Regularization methods may avoid overfitting in logistic models. Since selecting the penalty parameter values plays an important role in the prediction performance of such models (e.g., the ridge, lasso and elastic net), one may ask whether choosing an appropriate optimality measure (used in cross-validation of the penalty parameters) will boost the models' prediction performance.

The main objective of this thesis is then to study the effect of different optimality measures (used in cross-validation of the penalty parameters) on the prediction performance of the penalized logistic regression methods, and to identify the appropriate and inappropriate performance measures for imbalanced data, in the setting of fraud detection.

On the one hand, we will incorporate the measures of prediction performance into the automated tuning procedures (e.g., cross-validation) when estimating the model. On the other hand, different measures of prediction performance will be explored in the validation phase. This will be conducted on simulated data and real credit card default data, which are class imbalanced and high-dimensional with a mixture of discrete and continuous covariates. In the simulation study, the Gaussian copula and inversion sampling will be used to generate such data, with arbitrary marginal distributions and a dependence structure.

The rest of the thesis is organized as follows. In Chapter 2, we describe the statistical methods for this study. Chapter 3 presents the methods for data simulation and experimental results of the simulation study. In Chapter 4, we present an illustration on a real credit card default data. Chapter 5 presents the conclusion of our study.

Chapter 2

Methods

2.1 Fraud detection problem

Fraud detection is an important task in many fields, including insurance, banking, tax administration, etc. In practice, it may refer to detection of fraudulent insurance claims, money laundering, tax fraud, etc. It is usually infeasible to manually investigate all the cases (i.e., insurance claims, transactions, tax reports and the like), since the total number of cases is typically very high. Also, such investigations are costly. In order to identify those cases that are most likely to be fraudulent, good prediction models are therefore required.

2.1.1 Class imbalanced and high-dimensional fraud data

Among the total cases, fraudulent activities occur rarely, so only a small proportion of the cases are fraudulent, which results in class imbalanced data. The proportion of the fraudulent cases is usually less than 2%. Under some circumstances, this proportion may even be less than 1%. Further, fraud data may also be high-dimensional, meaning the number of covariates is relatively large in comparison to the number of cases. To be specific, the number p of covariates is often approximately the same as the number n of cases, or sometimes, p is only slightly smaller than n. Additionally, the data registered on each case usually contain many types of information, resulting in a mixture of continuous (e.g., income) and discrete (e.g., gender) variables. There may exist dependence between the covariates, and some of the covariates may be irrelevant to the outcome.

2.1.2 Problems and possible solutions

The rare or fraudulent cases, which we are mainly interested in detecting, have proven to be difficult to detect (Krawczyk, 2016; Weiss, 2004), and class imbalance may cause suboptimal classification performance (Chawla et al., 2004). Traditional machine learning methods may assume or expect a somewhat balanced class distributions in the data (He and Edwardo A, 2009). For class imbalanced data, they may perform poorly on the minority class. One reason may be that the models simply get too little information about the minority class, which is often underrepresented and lacks a clear structure (Krawczyk, 2016). Re-sampling methods directly transform the training data into more class balanced data (not necessarily fully balanced) before building the model. More details of re-sampling methods (e.g., oversampling) are given in Section 3.5.

Meanwhile, class imbalanced data are not always difficult to learn, for instance, when the minority cases are not too few and there is little degree of overlapping among the classes in the data (Batista et al., 2004). That is, however, untypical. Class overlapping is common, and highly overlapping classes can impair model learning and model performance substantially. Data cleaning methods using Tomek links, may relieve the overlapping among the classes by removing the noisy and borderline cases. This is also presented in Section 3.5. It is worth noting that the aforementioned methods in Section 3.5 are only discussed, but not applied in this thesis, since we study mainly the measures of prediction performance for class imbalance in the setting of fraud detection.

Typical measures of prediction performance, such as the accuracy, the AUC and the Brier score, are dominated by the majority class, thus, the model may still give quite good results, in spite of its poor prediction performance on the minority class. Consequently, wrong conclusions about the prediction performance of a model may be drawn, and suboptimal models may be chosen in model selection. Model selection here refers to the whole procedure of estimating the performance of different models in order to choose the best one (Hastie et al., 2009), including building the model and selecting the best tuning parameters. For class imbalance, more appropriate evaluation metrics should be applied in order to correctly evaluate the prediction performance, both in the training phase and in the validation phase. Several evaluation metrics (e.g., the AUCPR and the AUC) are discussed and compared in Section 2.6.

In addition, regular methods applied to high-dimensional data can lead to overfitting along with high variance. Regularization methods may be used to avoid overfitting, and they are given in Section 2.4. Variable selection applied before building the model may be carried out to select a subset of covariates that are most informative, in an attempt to achieve the optimal prediction performance (Haixiang et al., 2017). This is, however, not considered in this thesis, since the selection procedure may be computationally

infeasible for a large number of covariates, and is extremely variable due to its inherent discreteness (Hastie et al., 2009; Zou and Hastie, 2005). Meanwhile, it is worth noting that some regularization methods (e.g., the lasso and elastic net) do automated variable selection, in the sense that they do not include all the covariates in the model.

2.2 Logistic regression

This section describes the logistic regression method.

2.2.1 Model presentation

Let Y denote a binary response variable. Given n observations, the outcome value of the ith observation is denoted by y_i, where $y_i \in \{0, 1\}$, and $y_i = 1$ represents that the ith observation is fraudulent. The corresponding p covariate values for each y_i are denoted by $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})$. The logistic regression method models the relationship between the binary outcome Y and the corresponding covariates $x_{i1}, \ldots, x_{ip}$ as

$$P(Y_i = 1|\boldsymbol{x}_i) = \frac{exp(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})}{1 + exp(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})}. \tag{2.1}$$

Here, $P(Y_i = 1|\boldsymbol{x}_i)$ is the probability of fraud in the ith observation, given information of the covariates. It is also the mean μ_i of the binary response $Y_i|\boldsymbol{x}_i$, which follows a Bernoulli distribution, i.e., $Y_i|\boldsymbol{x}_i \sim Bernoulli(p(\boldsymbol{x}_i))$, where $p(\boldsymbol{x}_i) = P(Y_i = 1|\boldsymbol{x}_i)$. The regression coefficients are denoted by $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)$, where β_0 is the intercept. The sigmoid structure of Equation (2.1) keeps the probability $p(\boldsymbol{x}_i)$ from falling outside the range $(0, 1)$, while the p covariates can take values over the entire real line. The Equation (2.1) can be reformulated as

$$\log\left(\frac{p(\boldsymbol{x}_i)}{1 - p(\boldsymbol{x}_i)}\right) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}, \tag{2.2}$$

where we observe the *logit* link function $g(\mu_i) = \log\left(\frac{\mu_i}{1-\mu_i}\right)$ on the left side of Equation (2.2). On the right side of Equation (2.2) is the linear predictor $\eta(\boldsymbol{x}_i)$, which represents a linear combination of the covariates. Thus, the decision boundary that separates the observations into two classes is linear for the logistic regression classifier. A natural choice in many settings, for instance, for class balanced data, is to have a threshold of 0.5 for converting predicted event probabilities produced by the logistic regression models into binary labels, meaning that the ith observation is predicted to be fraudulent (i.e., $\hat{y}_i = 1$) if its predicted event probability

$$\hat{p}(\boldsymbol{x}_i) > 0.5. \tag{2.3}$$

We therefore that

$$\log\left(\frac{\hat{p}(\boldsymbol{x}_i)}{1-\hat{p}(\boldsymbol{x}_i)}\right) > 0.$$

Equivalently, according to Equation (2.2), $\hat{y}_i = 1$ if the linear predictor

$$\eta(\boldsymbol{x}_i) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} > 0.$$

However, for class imbalanced data, it does not have to be like that, and the threshold could be fixed to the ratio of class imbalance, or may be optimized according to different evaluation metrics. Further details related to the threshold are discussed in Section 2.6.

2.2.2 Maximum likelihood estimation

In order to estimate the parameters $\boldsymbol{\beta}$, we usually fit logistic regression models with the maximum likelihood estimation method. Given n data points that are independent and Bernoulli distributed (i.e., $Y_i|\boldsymbol{x}_i \sim Bernoulli(p(\boldsymbol{x}_i))$), the likelihood function is given by

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} p(\boldsymbol{x}_i)^{y_i}(1-p(\boldsymbol{x}_i))^{1-y_i}. \tag{2.4}$$

The corresponding log-likelihood function is then

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[y_i \log\left(p(\boldsymbol{x}_i)\right) + (1-y_i)\log\left(1-p(\boldsymbol{x}_i)\right)\right],$$

which can be reformulated as

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) - \log(1 + e^{\beta_0+\beta_1 x_{i1}+\cdots+\beta_p x_{ip}})\right]. \tag{2.5}$$

Consequently, the maximum likelihood estimate is given by

$$\hat{\boldsymbol{\beta}}^{MLE} = \underset{\beta_0,\beta_1,\ldots,\beta_p}{argmax} \left\{ \sum_{i=1}^{n} \left[y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) - \log(1 + e^{\beta_0+\beta_1 x_{i1}+\cdots+\beta_p x_{ip}})\right] \right\}. \tag{2.6}$$

To determine the $\hat{\boldsymbol{\beta}}^{MLE}$ values at which the log-likelihood function $l(\boldsymbol{\beta})$ is maximized, we set the derivatives of Equation (2.5) to zero and obtain

$$\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} \boldsymbol{x}_i \left(y_i - p(\boldsymbol{x}_i)\right) = \mathbf{0}, \tag{2.7}$$

which are $p+1$ non-linear equations in $\boldsymbol{\beta}$ (through $p(\boldsymbol{x}_i)$). The solutions may be determined by, for instance, *iteratively reweighted least squares* (Hastie et al., 2009), which is implemented in the R function **glm**. The estimated probability of fraud, given covariate information $\boldsymbol{x} = (x_1, \ldots, x_p)$, is then modelled by

$$\hat{p}(\boldsymbol{x}) = \frac{exp(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p)}{1 + exp(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p)}. \tag{2.8}$$

2.2.3 Deviance

A relevant term here is the deviance (Agresti, 2015; Hosmer Jr et al., 2013), which is given by

$$deviance = -2 \cdot (l(\boldsymbol{\beta}) - l^{\star}) .$$

Here, $l^{\star}$ is the log-likelihood of the saturated model. If we treat the likelihood $L(\boldsymbol{\beta})$ in Equation (2.4) as a function of $p(\boldsymbol{x}_i)$, for $i = 1, \ldots, n$, the saturated model should achieve the maximum likelihood with the perfect fit $\hat{p}(\boldsymbol{x}_i) = y_i$, for $i = 1, \ldots, n$. The likelihood of the saturated model is then given by

$$L^{\star} = \prod_{i=1}^{n} y_i^{y_i}(1 - y_i)^{1-y_i} = 1,$$

and then the log-likelihood of the saturated model

$$l^{\star} = 0.$$

Hence, the deviance of a fitted logistic regression model is given by

$$deviance = -2 \cdot l(\boldsymbol{\beta}). \tag{2.9}$$

We can compare the deviance of two models in order to check if one gives a better fit than the other. Here, to minimize the deviance is equivalent to maximize the likelihood. The deviance may be used as an optimality measure in cross-validation (see Section 2.5) for determining the optimal values of tuning parameters.

2.2.4 Extension and potential problems

The logistic regression model can be used for interpretation of the role of covariates in explaining the outcome and prediction. It is commonly used as a benchmark in empirical studies and as a prediction tool in practice (e.g., credit card default). Alternatively, for $j = 1, \ldots, p$, a non-linear function $f(x_j)$ (e.g., a quadratic term) may be implemented on the right side of Equation (2.2), if the non-linear effects are more informative about the outcome and improve the predictive power. Interaction terms may also be added in the model. However, they are unknown and must be found, which is difficult. For data with p covariates, the number of possible interaction terms is $\frac{p \cdot (p-1)}{2}$, which can become very large for a big p (e.g., $p = 1000$). Also, for high-dimensional data, it might not be feasible or desirable to have more terms included in the model, as the maximum likelihood estimation method may fail in high-dimensional situations with large numbers of parameters (Efron and Hastie, 2021). Further, maximum likelihood may suffer in case of complete separation where, for instance, a covariate perfectly separates the two classes, and the model may not converge using the R function **glm**. Other concerns

with high-dimensional data are high variance in the maximum likelihood estimates and overfitting.

Therefore, we turn to regularization approaches that can overcome the dimensionality problem. To be specific, we choose penalized logistic regression models with constraints imposed on the regression coefficients, or equivalently, with penalty terms attached to the log-likelihood function, to handle class imbalanced and high-dimensional data within fraud detection.

2.3 The bias-variance trade-off and model complexity

Let $\boldsymbol{X} \in \mathbb{R}^p$ denote a p dimensional random input vector, and $Y \in \mathbb{R}$ a random output variable. The expected prediction error is given by

$$Err = E[L(Y, \hat{f}(\boldsymbol{X}))], \tag{2.10}$$

where $\hat{f}$ is the fitted model, and $L(Y, \hat{f}(\boldsymbol{X}))$ is an adequate loss function for measuring errors between the outcome Y and the predicted outcome $\hat{f}(\boldsymbol{X})$.

To select the right prediction model out of many candidates, is basically to find a model which minimizes the prediction error with some in-between model complexity. Due to the trade-off between bias and variance, the prediction error will not always decrease like the training error does, as the model complexity increases. This as can be seen in Figure 2.1 from Hastie et al. (2009). The reason is that the training error is the prediction error obtained by applying the fitted model to the same data used for fitting it, while the prediction error is obtained on new data. The training error is often lower than the prediction error, since the fitted model has already seen the training sample. The training error may decrease to zero, as the model complexity increases enough. The prediction error is related to whether the model will generalize well. We want to minimize the prediction error, such that the model has good enough prediction performance on new data.

Generally, the minimization of prediction error should take both the bias and the variance into consideration as the model complexity varies. The bias and variance typically act in the opposite way, and their behavior is strongly connected to the model complexity. A more complex model tends to have smaller bias and larger variance, while it is the other way around for a simpler model. An overfitted model with high variance captures the noise along with the underlying pattern in the training data, and hence, it may not generalize well. On the contrary, an underfitted model with large

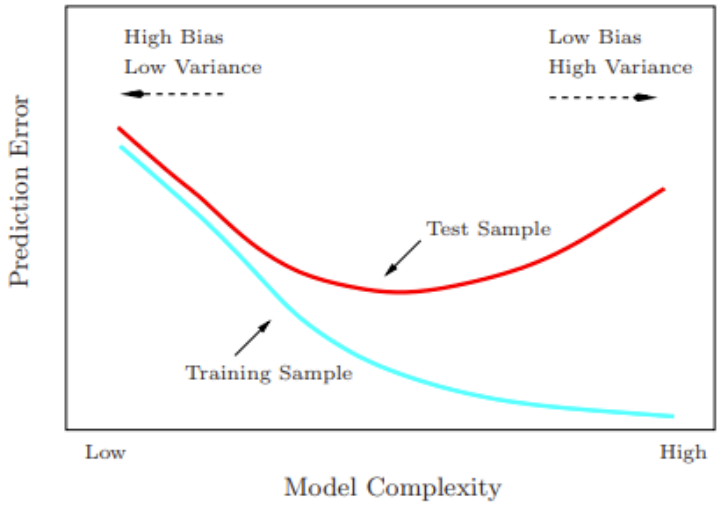


Figure 2.1: Test and training error as a function of model complexity.

bias is not complex enough. Though with a low variance, i.e., a low uncertainty, it may not represent the true underlying relation between covariates and outcome, and hence, it may not generalize well, either. Overall, it is desired to build a neither overfitted nor underfitted model. The problem of finding the right model is then finding the intermediate point with the right balance between the bias and variance, in other words, with the right model complexity.

2.4 Regularization methods

This section presents the lasso, ridge and elastic net for the logistic regression method.

2.4.1 The lasso

The lasso is a regularization method with the L_1-norm penalty $\lambda \sum_{j=1}^{p} |\beta_j|$ (Tibshirani, 1996). Given n observations with inputs $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$, for $i = 1, 2, \ldots, n$, the lasso solution for the logistic regression method is given by

$$\hat{\boldsymbol{\beta}}^{lasso} = \underset{\beta_0, \beta_1, \ldots, \beta_p}{argmax} \left\{ \sum_{i=1}^{n} \left[y_i(\eta(\boldsymbol{x}_i)) - \log(1 + e^{\eta(\boldsymbol{x}_i)}) \right] - \lambda \sum_{j=1}^{p} |\beta_j| \right\}, \tag{2.11}$$

where $\sum_{i=1}^{n} \left[y_i(\eta(\boldsymbol{x}_i)) - \log(1 + e^{\eta(\boldsymbol{x}_i)}) \right]$ is only the log-likelihood function $l(\boldsymbol{\beta})$ in Equation (2.5). Notice that the intercept term β_0 is not considered in the regularization. It is not a convention to shrink a parameter whose estimate gives the reference level of

log-odds. Also, for $\lambda \to \infty$, the intercept estimate is given as

$$\hat{\beta}_0 = \log\left(\frac{\bar{y}}{1-\bar{y}}\right),$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$. The shrinkage parameter λ ($\lambda \geq 0$) controls the magnitude of penalization. When $\lambda = 0$, there is no penalty, thus, the maximum likelihood estimate $\hat{\boldsymbol{\beta}}^{MLE}$ is obtained. The larger the value of λ is, the more the coefficients are shrunk towards zero. Further, with a sufficiently large λ, some of the coefficients are set exactly to zero, due to the L_1-norm penalty structure with non-differentiable corners. Therefore, the lasso does some sort of variable selection and produces simpler and sparser models than the unrestricted maximum likelihood. This shrinkage effect of the lasso is highly effective, when there are a large number of covariates and only a few are expected to have a considerable effect on modeling outcomes. Compared to the unrestricted logistic regression model, the less complex model by the lasso is generally easier to interpret and use in practice.

The penalty introduces some bias to the coefficient estimates, but it also provides more stable estimates with smaller variance (Le Cessie and Van Houwelingen, 1992). Hopefully, this will improve the prediction performance, provided with a good choice of λ value. Deciding the optimal value of λ becomes a trade-off between bias and variance, which was previously discussed in Section 2.3. Here, λ is also a tuning parameter. It is a common choice to adaptively choose the λ value by cross-validation discussed in Section 2.5, such that the estimated expected prediction error is minimized. In other words, the optimal value of λ gives theoretically the best expected prediction performance.

2.4.2 Ridge regression

Ridge regression is a regularization method like the lasso, but with the L_2-norm penalty $\lambda\sum_{j=1}^{p}|\beta_j|^2$ (Hoerl and Kennard, 1970). The ridge estimate for the logistic regression method is given by

$$\hat{\boldsymbol{\beta}}^{ridge} = \underset{\beta_0,\beta_1,\ldots,\beta_p}{argmax}\left\{\sum_{i=1}^{n}\left[y_i(\eta(\boldsymbol{x}_i)) - \log(1+e^{\eta(\boldsymbol{x}_i)})\right] - \lambda\sum_{j=1}^{p}|\beta_j|^2\right\}. \tag{2.12}$$

As in the lasso, the intercept term is not involved in the penalization. With $\lambda = 0$, the maximum likelihood estimate $\hat{\boldsymbol{\beta}}^{MLE}$ is obtained. The shrinkage parameter λ should also be tuned, in order that the fitted model is neither underfitted nor overfitted. Unlike the lasso penalty, the ridge penalty is now quadratic and differentiable at zero. As the shrinkage parameter λ increases, the larger penalty will shrink all the coefficients toward zero proportionally, but no coefficient will be exactly zero. All the covariates

are included in the model. The ridge regression does not do variable selection, but it handles correlated covariates well, as the ridge penalty shrinks the coefficients of correlated covariates towards each other (Hastie et al., 2021). The lasso, on the other hand, can be a little chaotic in handling correlation in variables, as some coefficients may enter, exit and reenter the model for different values of the penalty parameter λ. The aforementioned shrinkage behavior of the lasso and ridge logistic regression can be observed in Figure A.2 and Figure A.1, respectively (in Appendix A.1).

Which of the lasso and ridge regression is better, depends on the modeling task and the type of data. The Lasso may be preferred over the ridge for interpretability reasons, since the former selects variables. It is worth noting that regularized approaches that do automated variable selection should, however, be used with care for interpretation or inference, due to the fact that they introduce a bias in the coefficient estimates. Instead, they are usually applied for the interest of prediction. For data with highly correlated covariates, the ridge, which is better at handling correlated covariates, may perform better than the lasso in prediction. Meanwhile, according to Hastie et al. (2009), it is generally safer to use lasso rather than ridge regression due to the *bet on sparsity* principle, especially when it is uncertain whether there exist many covariates with small effect, or a few covariates with large effect on the outcome. In the former situation, the ridge performs only slightly better, and the lasso may still not perform much worse. However, in the latter situation, the lasso may perform much better than the ridge, since the L_1 penalty is better suited to sparse situations (Hastie et al., 2009).

2.4.3 The elastic net

The elastic net penalty introduced by Zou and Hastie (2005) is a compromise solution between the lasso and ridge regression. It has both the L_1-norm penalty and the L_2-norm penalty. The elastic net regression estimate for the logistic regression method is given by

$$\hat{\boldsymbol{\beta}}^{elnet} = \underset{\beta_0,\beta_1,\ldots,\beta_p}{argmax} \left\{ \sum_{i=1}^{n} \left[y_i(\eta(\boldsymbol{x}_i)) - \log(1 + e^{\eta(\boldsymbol{x}_i)}) \right] - \lambda \sum_{j=1}^{p} (\alpha|\beta_j| + (1-\alpha)|\beta_j|^2) \right\}, \tag{2.13}$$

where $\alpha \in [0, 1]$. Here, α controls the balance between the L_1-norm and L_2-norm penalty, and the intercept term is not involved in the penalization. For $\alpha = 0$, the ridge is given, and $\alpha = 1$ gives the lasso. The α is a tuning parameter, which must be found in addition to λ. The optimal value of α may be found through a grid search, which is a time consuming task. The extra computational cost is one disadvantage with the elastic net penalty. Meanwhile, the elastic net is supposed to enclose the advantages of both penalties. On the one hand, the elastic net penalty has non-differentiable corners due to the L_1-norm penalty, so it selects variables like the lasso and produces sparse

models. On the other hand, it handles correlated covariates like the ridge, through shrinking together the coefficients of correlated covariates.

2.4.4 Standardization

The ridge, lasso and elastic net regression need initial standardization of the covariates, such that the penalization is fair to all covariates (Tibshirani, 1997). Without standardization, the penalty is dependent on the scaling of the covariates. For instance, we assume that one covariate x_1 is in a very large scale of thousands while another covariate x_2 varies from 0 to 1, and neither of them is standardized. Their coefficients β_1 and β_2 are then not on the same scale, and probably $|\beta_1| << |\beta_2|$. Hence, the same regularization λ would likely have less effect on x_1 than on x_2, which is not desired. The standardization centers and scales the inputs, such that the transformed inputs have mean zero and unit variance. It is carried out before solving Equation (2.11), Equation (2.12) and Equation (2.13). Each x_{ij} in the inputs is standardized into $x_{ij}^{'}$ by

$$x_{ij}^{'} = \frac{x_{ij} - \bar{x}_j}{s_j} \tag{2.14}$$

with $\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$ and $s_j = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2}$.

2.5 Cross-validation

With sufficient data, it is possible to split the available data into three parts, i.e., the training set, validation set and test set. The training set is used to train the models, and the validation set is used to estimate the prediction performance for different models. Both data sets serve for the goal of model selection, that is, estimating the performance of different models in order to find the best one with minimum prediction error (Hastie et al., 2009). After having chosen the best model, the test set is used for estimating its prediction performance. It is important that the test set is kept aside from model selection and taken out only for model assessment.

However, in reality, it may happen that we are not in a data-rich situation, and the data cannot be split into three sufficient parts. Assume, in this thesis, that we are in the situation where there are insufficient data, the data are only abundant enough to be split into a training set and a test set. An efficient sample reuse method, i.e., cross-validation is used in model selection, not only to estimate the averaged prediction error, but also to select the best values of tuning parameters. Therefore, cross-validation is applied to the training set as part of the training process. We present cross-validation and one slightly altered version, i.e., stratified cross-validation. Both methods estimate the

expected prediction error given in Equation 2.10, and they are applied to the training set.

2.5.1 Regular k-fold cross-validation

To perform cross-validation, the training set is randomly divided into k folds, and each fold has approximately the same amount of observations. A learning method is, in turn, fitted to $k-1$ folds, and evaluated on the remaining one fold. As a result, there are k fitted models, and each model gives a prediction error estimate. By taking the average of these prediction error estimates, we obtain an estimated expected prediction error

$$CV(\hat{f}) = \frac{1}{k}\sum_{i=1}^{k} L(\hat{f}^i, (y, \boldsymbol{x})_{I_i}).$$

Here, $L(\hat{f}^i, (y, \boldsymbol{x})_{I_i})$ represents the prediction error estimate of the ith model fit $\hat{f}^i$, which is trained on the training set without the ith fold, and evaluated on the remaining ith fold. With the training set denoted by $(y, \boldsymbol{x})$, the observations in the remaining ith fold are represented by $(y, \boldsymbol{x})_{I_i}$, where I_i consists of indices of elements in the ith fold.

In order to reduce the randomness in the splitting of the training set into k folds, we can use repeated cross-validation, where we repeat the splitting of the training set m times, and in the end, take the average of m estimated expected prediction errors, i.e., $CV^1(\hat{f}), \ldots, CV^m(\hat{f})$.

For shrinkage methods (see Section 2.4), the problem of deciding one or several best tuning parameters, may arise. The values of the tuning parameters vary the model complexity, which is associated with the bias-variance trade-off (see Section 2.3). For a learning method with shrinkage, finding the best model complexity is concerned with finding the optimal values of the tuning parameters, which may be chosen based on cross-validation. We perform cross-validation for a range of values of one tuning parameter, or for a grid of values of two tuning parameters, depending on the shrinkage method. An estimated expected prediction error curve may be obtained as a function of the tuning parameters. We choose the values of the tuning parameters that minimize the estimated expected prediction error.

2.5.2 Stratified k-fold cross-validation

Regular k-fold cross-validation, where the training data are randomly split into k subsets, does not consider whether the distribution of the binary output in each fold is approximately the same as in the original training set. It is very likely that some subsets may contain fewer positive cases than they should, with respect to the proportion of

positive cases in the original data. Moreover, when the whole training data have very few positive cases, some subsets may contain no positive cases at all.

Unlike regular cross-validation, stratified cross-validation makes sure that every fold has roughly the same class distribution as the whole training set. This is important for class imbalanced data, where the number of negative cases greatly exceeds the number of positive cases. Therefore, it is more appropriate to apply stratified cross-validation to class imbalanced data.

A general algorithm 2.1.1 for performing either regular cross-validation, or stratified cross-validation, is given below, including the steps for computing prediction errors for a range of values of one tuning parameter λ. As we can see in Algorithm 2.1.1, the main difference between performing regular cross-validation and stratified cross-validation is that, before sampling random labels for each observation, the former does not separate the two classes, whereas the latter separates two classes, in order to achieve approximately the same proportion of classes in each fold.

2.5.3 The choice of k

While it is clear that stratified cross-validation suits class imbalanced data, the number k of folds remains to be determined. The choice of k is related to both the computational complexity, and the typical bias and variance trade-off.

It is known that cross-validation is time consuming. For instance, the computational complexity of k-fold cross-validation, is k times that of training the algorithm on the whole training set once. For real fraud detection problems, where thousands of new data come in constantly, it is necessary to give thought to the total time it costs, for a prediction method to learn from the historical data and to produce reliable results.

Further, as fraudsters constantly adapt themselves to prevention and detection technologies, fraud detection models need also to be adaptive and updated at fixed time points, or continuously over time (Bolton and Hand, 2002). Thus, the time it takes for the learning models to be constructed, or evolve, should preferably be rather short. Therefore, the size of k should preferably be small, since the computational complexity is roughly proportional to the number of data splits (Arlot and Celisse, 2010).

With a smaller k, the training folds contain less observations, and they will be more different from the whole training set. Thus, it will lead to larger bias and lower variance in the whole cross-validated estimator of expected prediction error, where the lower variance is due to the effect of averaging.

Algorithm 2.1.1 K-fold cross-validation

Input: $\boldsymbol{y} = (y_1, \ldots, y_n), k, method, I = \{1, \ldots, n\}, \lambda_1, \ldots, \lambda_m$

if (method = regular) **then**
  Sample I_1 from I without replacement
  for $i = 2, \ldots, k$ **do**
    Sample I_i from $I \setminus \bigcup_{j=1}^{i-1} I_j$ # I_i consists of indices of elements in fold i
  end for
else if (method = stratified) **then**
  Generate $I_1 = \{i : y_i = 1\}$ # I_1 consists of indices of positive cases
  Generate $I_0 = \{i : y_i = 0\}$ # I_0 consists of indices of negative cases
  Sample I_{11} from I_1 without replacement
  Sample I_{01} from I_0 without replacement
  for $i = 2, \ldots, k$ **do**
    Sample I_{1i} from $I_1 \setminus \bigcup_{j=1}^{i-1} I_{1j}$ # I_{1i} consists of indices of positive elements in fold i
    Sample I_{0i} from $I_0 \setminus \bigcup_{j=1}^{i-1} I_{0j}$ # I_{0i} consists of indices of negative elements in fold i
  end for
end if
for $l = 1, \ldots, m$ **do**
  for $i = 1, \ldots, k$ **do**
    Fit the model to folds $I \setminus I_i$ and obtain a model fit $\hat{f}^i(\lambda_l)$
    Compute the loss $L\left(\hat{f}^i(\lambda_l), (y, \boldsymbol{x})_{I_i}\right)$ # $(y, \boldsymbol{x})_{I_i}$ are observations in fold i
  end for
  Let $CV(\hat{f}, \lambda_l) = \frac{1}{k} \sum_{i=1}^{k} L\left(\hat{f}^i(\lambda_l), (y, \boldsymbol{x})_{I_i}\right)$
end for
Return: $CV(\hat{f}, \lambda_1), \ldots, CV(\hat{f}, \lambda_m)$

On the contrary, with a larger k, the training folds will consist of more observations, and they will be more similar to the whole training set. Hence, the whole cross-validated estimator will have smaller bias, but higher variance. The reason for higher variance is that the training folds will also be very similar to each other, and similar models will probably be produced over these similar folds. Therefore, the expected prediction error estimate will be very sample-specific for this particular data set. With a different data set from the same underlying distribution, one could obtain a very different estimate of the expected prediction error. For instance, when the number of folds k is equal to the number of observations in the training set, along with an excessively high computational complexity, leave-one-out cross-validation gives almost unbiased expected prediction error, but the whole cross-validation estimator can have very high variance.

In fact, Breiman and Spector (1992) have shown that five-fold (i.e., $k = 5$) cross-validation performs better than leave-one-out cross-validation at submodel selection and evaluation. Besides, five- or ten-fold cross-validation are recommended as a good compromise in Hastie et al. (2009). Moreover, Kohavi (1995) has conducted a study in comparing and evaluating the performance of cross-validation in accuracy estimation, with different numbers of folds and whether the folds are stratified or not. The study suggested the use of stratified cross-validation for model selection, since it improves the performance, in terms of achieving both lower bias and smaller variance. Also, stratified ten-fold (i.e., $k = 10$) cross-validation is recommended, even when more folds are computationally possible (Kohavi, 1995).

However, stratified cross-validation does not consider whether or not a balanced distribution in the feature space is maintained for each class. This may be a potential problem and may affect the quality of estimated error (Zeng and Martinez, 2000), if the feature distribution within each of the two classes consists of several clusters. Actually, a method, which is called distribution-balanced stratified cross-validation (DBSCV), is proposed and proven to improve the estimation quality in such cases (Zeng and Martinez, 2000). We assume that there is only one cluster per class in this thesis, so the distribution-balanced stratified cross-validation (DBSCV) method is not relevant.

2.6 Evaluation metrics

There are more than twenty measures to choose from when it comes to evaluating and comparing the predictive abilities of classifiers. It is not straightforward to directly draw conclusions on the most appropriate evaluation metrics for assessing the performance of classifiers, for severe class imbalance. In fact, the problem of choosing an evaluation metric is complicated, and solutions may be inconclusive, depending on the characteristics of data, and the specific objectives of prediction problems.

In this section, we study several evaluation metrics, compare and discuss their characteristics and limitations in measuring the prediction performance of classification models, especially in imbalanced learning. The three types of evaluation metrics are respectively threshold based metrics, ranking metrics and probabilistic metrics. Threshold based metrics are developed on the basis of a confusion matrix, which is defined by one threshold value. So the values of threshold based metrics vary over different thresholds. They are useful measures in evaluating the classification ability of a binary classifier. A binary classifier produces a class label, indicating the mostly likely class that a case should belong to. Ranking metrics evaluate the ability of a probabilistic classifier in ranking true positive cases higher than true negative cases. A probabilistic classifier assigns a case an estimated probability of class membership, rather than a crisp class label (i.e., positive or negative). For instance, a logistic regression model produces a predicted event probability $\hat{P}(Y = 1|\boldsymbol{x}) = \hat{p}(\boldsymbol{x}) \in (0, 1)$, given covariate information $\boldsymbol{x}$. Probabilistic metrics assess the accuracy of probabilistic predictions by checking the goodness of predicted probability scores. Evaluation metrics are evaluated on the test set.

2.6.1 Threshold based metrics

Assume that the test set consists of P positive cases and N negative cases. Let τ denote the ratio of class imbalance, which is defined as the proportion of positive cases among all the cases, i.e., $\tau = \frac{P}{P+N}$. We have $\tau = 0.5$ for class balanced data.

Probabilities transformed to binary labels

A probabilistic classifier can be easily transformed into a binary classification model, given a threshold value $t \in [0, 1]$. For instance, a logistic regression model produces a predicted event probability $\hat{p}(\boldsymbol{x})$, to which a certain threshold value t may be applied. The cases are predicted to be negative (or non-fraudulent) if $\hat{p}(\boldsymbol{x}) < t$, and positive (or fraudulent), otherwise. The transformation rule is given by

$$\hat{y} = \begin{cases} 0, & \hat{p}(\boldsymbol{x}) < t, \\ 1, & \text{otherwise.} \end{cases} \tag{2.15}$$

The positive and negative cases are respectively represented by 1 and 0. If the threshold value $t = 0$, all the cases are predicted to be positive, and if $t = 1$, all the cases are predicted to be negative. The threshold value t is usually set to be τ for evaluating and comparing the prediction performance of binary classifiers. In this thesis, we select threshold values that maximize the prediction performance of binary classifiers, Therefore, the optimal threshold value t^{opt} is flexible, and it depends on the ratio of class imbalance, the type of measures, the learning methods, etc.

Confusion matrix

For each case, the binary classifier can produce only four possible outcomes, that is, an actual positive case is classified to be positive (TP), an actual negative case is classified to be negative (TN), an actual positive case is classified to be negative (FN), and an actual negative case is classified to be positive (FP). It is practical to sum up all possible prediction outcomes of the whole test set in a confusion matrix. A generic confusion matrix for such binary classification problems is given in Table 2.1. It follows that

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

Table 2.1: A confusion matrix

$P = TP + FN$ and $N = TN + FP$. Threshold based measures are formed through linear or nonlinear combinations of the four numbers (i.e., TP, TN, FN and FP) from the confusion matrix. Their values depend on the threshold value t, since one threshold value determines one confusion matrix.

Accuracy

One typical threshold based metric is the overall prediction accuracy, or, equivalently, the misclassification rate (i.e., 1− accuracy). The overall prediction accuracy is the proportion of correctly classified cases in the data, and is given by

$$accuracy = \frac{TP + TN}{P + N}.$$

It evaluates a model's performance in predicting both classes correctly. It is an adequate measure in class balanced classification scenarios, where both the positive cases and the negative cases are equally important. However, when the data are highly class imbalanced, the accuracy will mainly measure how well the model predicts the majority class, as it outnumbers the minority class to a large extent, thus, the model may give overly optimistic results. As an example, we consider a test set with 100 positive cases and 900 negative cases, and the learner predicts all the cases to be negative. Although the accuracy is 0.9, which seems pretty good, this model does not correctly predict rare events at all. In this case, the accuracy metric is not effective, considering that our main interest within fraud detection lies in correctly predicting the true positives. Therefore, in class imbalanced classification scenarios, using accuracy as the only performance measure may lead to suboptimal classification models.

TPR, TNR and FPR

The sensitivity and specificity evaluate respectively the prediction performance in correctly detecting the positives and negatives. Actually, the sensitivity and specificity are the accuracy calculated for the positive class and the negative class, respectively. They are also called the true positive rate (TPR) and the true negative rate (TNR), respectively, and are given by

$$sensitivity = \ true\ positive\ rate = \frac{TP}{P},$$
$$specificity = \ true\ negative\ rate = \frac{TN}{N}.$$

The 1-specificity, i.e., the false positive rate (FPR) is also an often used metric. It equals $1 - TNR$, which leads to

$$1 - specificity = \ false\ positive\ rate = \frac{FP}{N}.$$

Obviously, there exists an inversion relation between the TNR and the FPR. The TPR, TNR and FPR are dependent on the threshold value t. Using the prediction rule (2.15), as t decreases, more cases will likely be assigned to be positive, thus, resulting in lower TNR, higher TPR and FPR. This can be seen in Figure 2.2. Although it is desirable to

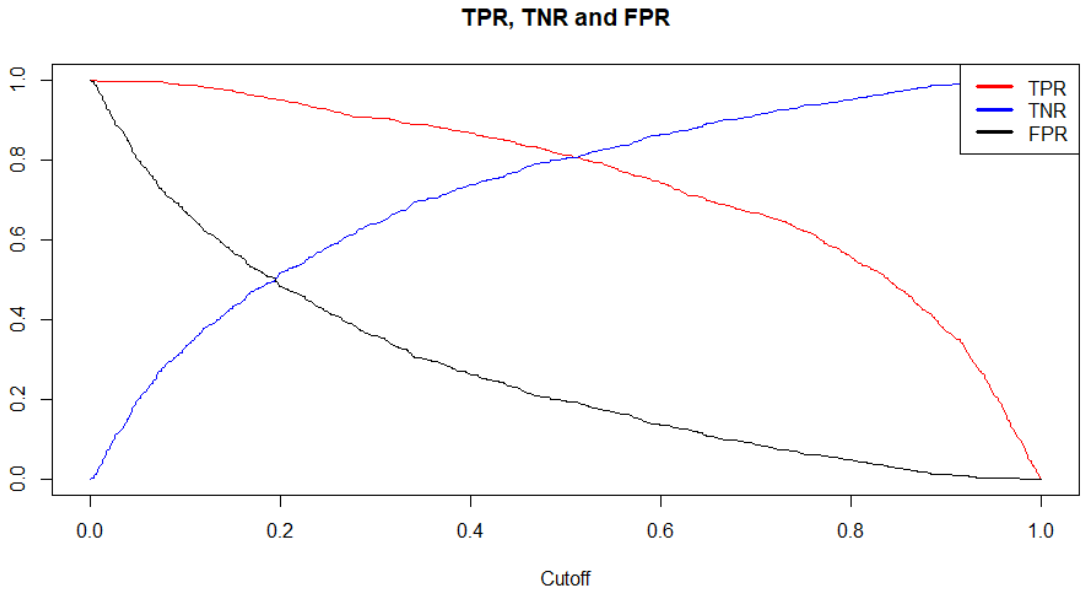


Figure 2.2: The TPR, TNR and FPR vary over all thresholds.

have higher TPR and TNR simultaneously, it is usually infeasible. In order to get a higher TPR, it needs to sacrifice some part of the TNR.

Gm and AGm

As there exists a trade-off relationship between the TPR and TNR, other evaluation metrics, which monitor the TPR and TNR at the same time, may be used. We consider the geometric mean and the adjusted geometric mean of the TPR and TNR, denoted by Gm and AGm, respectively. The Gm is given by

$$Gm = \sqrt{sensitivity \cdot specificity} = \sqrt{TPR \cdot TNR},$$

which gives equal weight to the changes in both the TPR and the TNR, so it should produce a relatively balanced combination of the TPR and the TNR. The Gm suits balanced class learning, where the TPR could be increased without worrying too much about the decrease in the TNR, in other words, a certain amount of increase in the FPR is acceptable. However, in fraud detection, a small amount of increase in the FPR may correspond to a large increase in FP, which is not wanted, considering the cost of looking into these actual non-fraudulent cases. Therefore, in imbalanced class learning, the Gm may lead to suboptimal models (Batuwita and Palade, 2009).

The adjusted geometric mean (AGm) of the TPR and the TNR is proposed in Batuwita and Palade (2009), and is given by

$$AGm = \begin{cases} \left(Gm + TNR * (1 - \tau)\right) / (2 - \tau), & TPR > 0, \\ 0, & TPR = 0. \end{cases}$$

Here, τ is the ratio of class imbalance. Compared to the Gm, the AGm is more sensitive towards the changes in the TNR than towards the changes in the TPR. Moreover, the lower the value of τ is, which corresponds to a more severe class imbalance, the more sensitive the AGm is towards the changes in the TNR. The AGm aims at solving the downside of Gm used as the performance measure in imbalanced class learning. It suits the type of imbalanced classification problems, where we want to increase the TPR as much as possible, while keeping the decrease in the TNR to the minimum, that is, the FPR should increase as little as possible (Batuwita and Palade, 2009).

Figure 2.3 shows the trade-off between the TPR and TNR in the left column, and the corresponding Gm and AGm in the right column, for Model I (upper panel) and Model II (lower panel), respectively. Model I and Model II are fitted to class balanced and class imbalanced data with $\tau = 0.5$ and $\tau = 0.1$, respectively. In the lower panel, the red and black dashed lines refer to the optimal threshold values for achieving the highest Gm and AGm, The cutoff on the x-axis is the threshold. respectively.

The Gm and AGm plots of Model I are very similar, as one can see in the upper panel of Figure 2.3. It is sufficient to choose the Gm as the performance measure in class

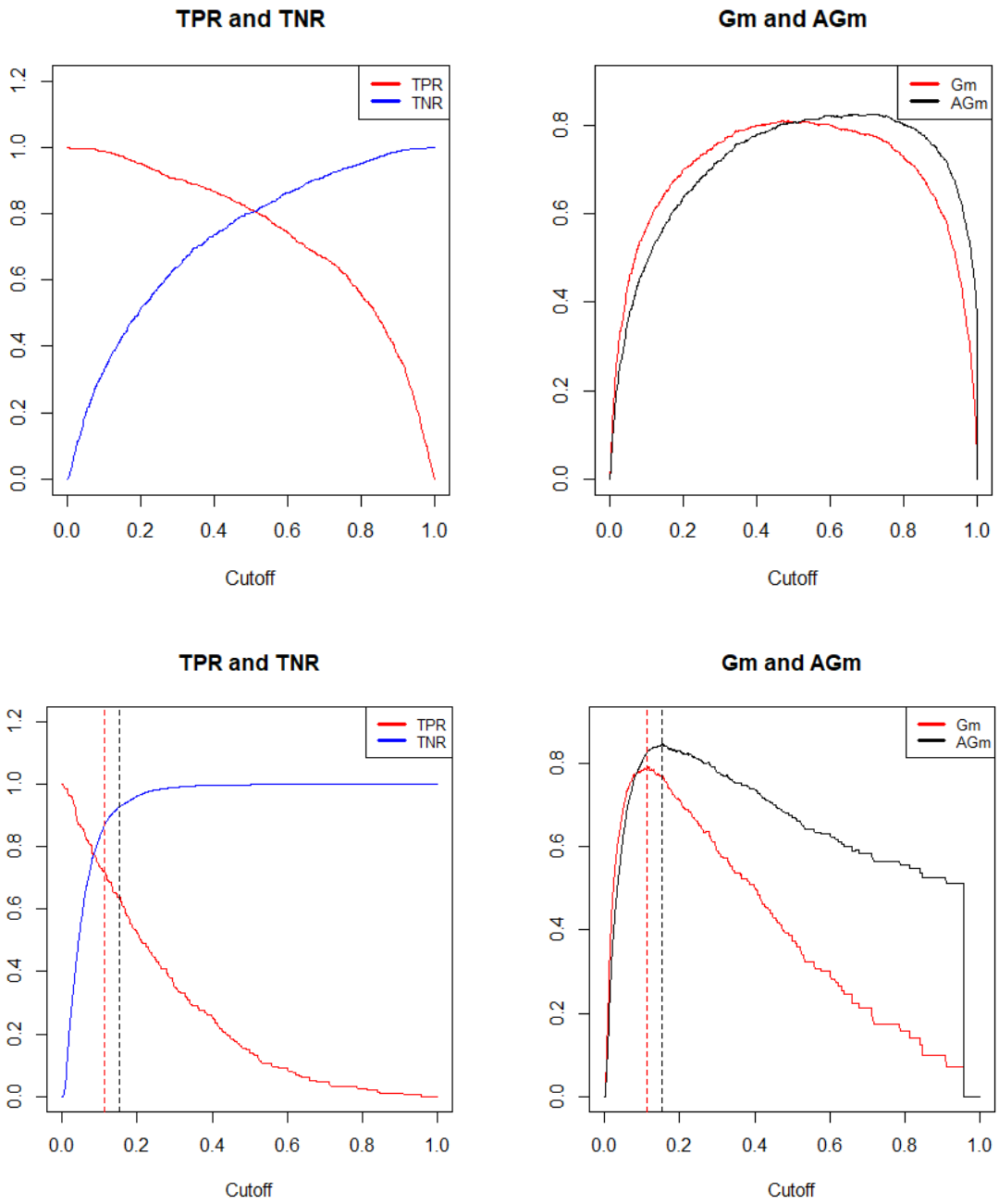


Figure 2.3: *Upper panel*: The TPR and TNR trade-off plot (*left*), and the corresponding Gm and AGm plots (*right*) for Model I. *Lower panel*: The TPR and TNR trade-off plot (*left*), and the corresponding Gm and AGm plots (*right*) for Model II.

balanced classification scenarios. Model I obtains the highest Gm at the threshold of 0.471, which is quite close to 0.5. For class imbalanced data, it indeed makes more sense to choose the Agm over the Gm. Model II achieves the highest Gm and AGm at thresholds of 0.114 and 0.154, respectively. Compared to the highest Gm, the highest AGm chooses a slightly higher threshold, which gives a lower TPR and a higher TNR.

Precision

Another threshold based metric, the precision, which is also referred to as the positive predictive value (PPV), assesses how precise a classifier is with its predictions. It is given by

$$precision = \frac{TP}{TP + FP}.$$

Intuitively, the precision measures how many of the predicted positive cases are truly positive, and it reflects the exactness of the prediction on the minority class.

The precision provides insight into how many cases are incorrectly assigned to the positive class through FP in the denominator. In fraud detection, it is important to have control over the size of FP. A rather big FP compared to TP gives a small precision value. If FP is much larger than TP, the cost in investigating the false alarms may exceed the gain from correctly detecting the true fraudulent cases, let alone the loss in not detecting some of the actual fraudulent cases. A big FP is not desired in fraud detection.

By checking the formula of the precision, it is clear that it is sensitive to class imbalance in the data. For highly class imbalanced data, even a small percentage change in the FPR corresponds to rather large changes in FP, and will lead to a distinctive change in the precision. By comparing FP to TP instead of N (i.e., the number of true negatives), the precision captures the effect of the large number of negative cases on the classifier's performance (Davis and Goadrich, 2006). Therefore, the precision could be a better metric for class imbalanced data than, for instance, the FPR (i.e., 1-TNR) and the accuracy. Both the FPR and the accuracy have N in the denominator, thus, the effect of large changes in FP would not obviously be observed in the FPR and the accuracy.

2.6.2 Ranking metrics

Ranking metrics measure how well a classifier orders the true positives over the true negatives based on the predicted class membership probability (Berrar, 2019). Unlike threshold based metrics, ranking metrics do not require a predefined threshold value. Before studying the behavior of such scalar measures, i.e., the AUC and the AUCPR, we start with the corresponding graphical tools, i.e., the receiver operating characteristics

curve (Fawcett, 2006; Swets, 1988) and the precision-recall curve (Davis and Goadrich, 2006; He and Edwardo A, 2009). The recall refers to the sensitivity, i.e., the true positive rate (TPR).

The receiver operating characteristics curve and the AUC

The ROC curve is the sensitivity (TPR) plotted against the 1-specificity (FPR) at all thresholds. Every point on the curve corresponds to a confusion matrix, defined by a threshold value. It is a useful visual tool for plotting the performance of classifiers. The more the ROC curve lies to the top left corner of the graph, the better predictive power the corresponding model has. Given the ROC curves of two classifiers, if the ROC curve of classifier I lies above or on the ROC curve of classifier II at every threshold, we say that classifier I dominates classifier II with better ranking performance. But if their ROC curves are intersected at some point, it is not easy to tell by eye which classifier is better on average. So it is generally impractical to use ROC plots for ranking the models from best to worst. Scalar metrics are more appealing.

The area under the ROC curve (AUC) is a summary measure acquired from the ROC curve. It tells us how good a classifier is in distinguishing one class from the other. It is shown that calculating the AUC is equivalent to computing the probability that a randomly chosen true positive case has a higher assigned score than a randomly chosen true negative case (Bradley, 1997; Burez and Van den Poel, 2009; Hanley and McNeil, 1982). The AUC value reveals the ranking ability of a model. A random guessing classifier has AUC = 0.5, and a perfect classifier has AUC = 1. Given that multiple classifiers are applied to the same test set, the larger the AUC value, the better the overall ranking ability of the corresponding model.

The precision-recall curve and the AUCPR

The precision-recall (PR) curve is constructed similarly as the ROC curve, but with the precision on the y-axis and the recall (TPR) on the x-axis. Unlike the ROC curve, the PR curve should lie close to the top right corner of the graph. A classifier's dominance is consistent in both ROC plots and PR plots. In other words, for a fixed number of positive and negative cases, a classifier's curve dominates other classifiers' curves in ROC space if and only if its curve dominates in PR space (Davis and Goadrich, 2006). However, a model that optimizes the area under the ROC curve is not guaranteed to optimize the area under the PR curve (Davis and Goadrich, 2006).

The area under the precision-recall curve (AUCPR), which is also referred to as the average precision, provides a summary metric for a classifier's performance. Both the precision and the recall only deal with the positive class. Therefore, the AUCPR

focuses mainly on the positive class. The higher the AUCPR value, the better the predictive performance on the minority class. However, unlike the AUC, the AUCPR is not guaranteed to be in $[0, 1]$, since there might be unreachable areas in PR space, depending on the ratio of class imbalance (Boyd et al., 2012). In PR space, a random guessing classifier is expected to have an AUCPR equal to the ratio τ of class imbalance (Saito and Rehmsmeier, 2015). For class balanced data, the baseline value of the AUCPR is 0.5. Hence, AUCPR = 0.35 indicates that the classifier performs much worse than a random guessing classifier, whereas if the proportion of positive cases is only 0.01 in the data, AUCPR = 0.35 does not necessarily indicate bad performance.

Comparison and contrast

The ROC curves and the AUC are not very sensitive to the changes in class imbalance, whereas the PR curves and the AUCPR may vary drastically as the ratio of class imbalance alters. This can been seen in Figure 2.4, where P and N are respectively the number of true positives and the number of true negatives in the test set. A learning method (i.e., the ridge logistic regression method) is respectively fitted to class balanced data with $\tau = 0.5$, and class imbalanced data with $\tau = 0.1$, thus, resulting in Model I (upper panel) and Model II (lower panel). They produce similar AUC values which are 0.894 and 0.852. The difference is only 0.042, which is insignificant. However, there are distinctive differences in terms of the PR curves and the AUCPR in the right column. Model I and Model II produce respectively the AUCPR values of 0.904 and 0.606. The AUCPR difference (0.298) is much larger than the AUC difference (0.042). The AUCPR, rather than the AUC, manages to distinctly reveal the fact that the learning method performs much better with class balanced data than with class imbalanced data.

The ROC curves and the AUC may give an overly optimistic view of a classifier's performance for tasks with highly class imbalanced data (Davis and Goadrich, 2006). The 1-specificity (FPR) on the horizontal line in ROC space, fails to reveal a decent change in FP, since the denominator N of FPR is often rather big for class imbalanced data with a majority of negatives. Consequently, over a wide range of thresholds decreasing from 1 towards 0, the FPR is almost unchanged or increases very slowly while the TPR increases, resulting in a nice ROC curve and a high AUC, despite many negatives being falsely classified as positives. The almost unchanged ROC curves and AUC values fail to capture the poor performance of the classifier for class imbalanced data. Using the two models from Figure 2.4, Figure 2.5presents how the TPRs and FPRs of Model I (upper panel) and Model II (lower panel) vary over all thresholds, respectively. The cutoff on the x-axis is the threshold. The bottom right plot in Figure 2.5 shows that the FPR of Model II nearly stays close to zero for a large range of thresholds from 1 to 0.2.

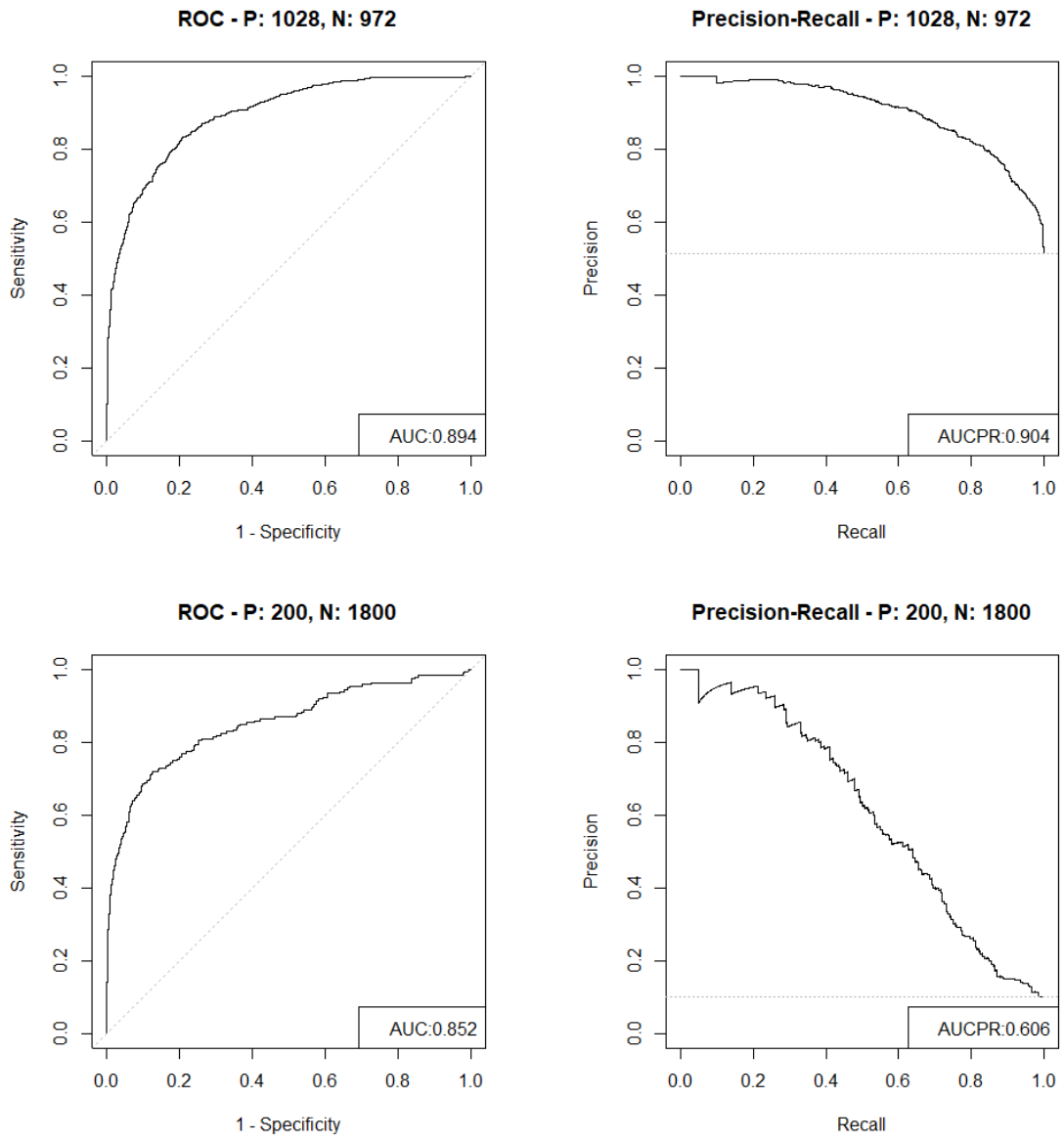


Figure 2.4: *Upper panel*: ROC and PR cuves for Model I. *Lower panel*: ROC and PR cuves for Model II.

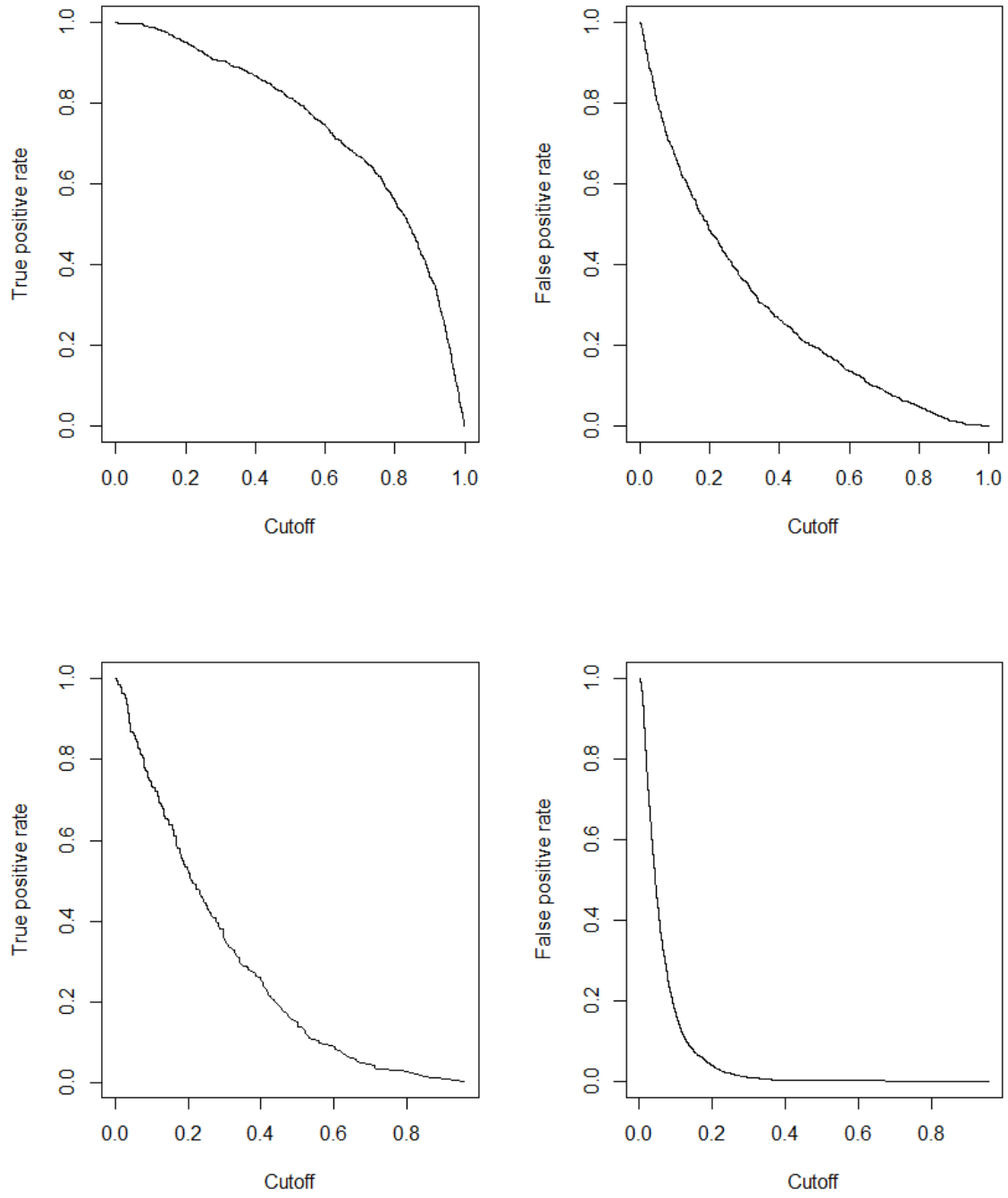


Figure 2.5: *Upper panel*: The TPR and FPR at all thresholds for Model I. *Lower panel*: The TPR and FPR for Model II.

The AUC, rather than the AUCPR, is more often used in comparing the performance of multiple classifiers. It is, however, insufficient to use only the AUC in class imbalanced learning scenarios, unless we do not take the misclassification cost of a large FP into consideration, and the objective of prediction is just to identify all the positive cases without worrying about the exactness of predictions. Both the positive class and the negative class should matter equally when using the AUC, which usually is not the case in fraud detection.

In summary, the PR curves and the AUCPR should be preferred over the ROC curves and the AUC, for tasks with severe class imbalance. The AUCPR, rather than the AUC, can effectively distinguish between a good and a very good model. Also, we usually care more about detecting the rare cases than the prevalent cases in fraud detection. The AUCPR measures the average precision, which is more informative than the AUC in accurately reflecting a classifier's prediction performance on the minority class.

2.6.3 Probabilistic metrics

Ranking metrics are more concerned with the relative ordering of cases, and focus on learning to distinguish between classes (Caruana, 2000). As a probabilistic metric, the Brier score is rather a measure of sharpness of a prediction model. The sharpness means specifically that the predicted event probabilities should be close to 1 for the true positives and to 0 for the true negatives, which is unnecessary for getting a high value of the AUC. A sharp prediction model means that it does not only distinguish well between the negative class and positive class, but also is quite confident about the predictions.

The Brier score (Brier, 1950) is an estimate of the true mean squared prediction error, and is given by

$$BS = \frac{1}{n}\sum_{i=1}^{n}(\hat{p}(\boldsymbol{x_i}) - y_i)^2,$$

where n is the number of observations in the test set, and $\hat{p}(\boldsymbol{x_i})$ is the predicted event probability for the ith observation.

We say that the $\hat{p}(\boldsymbol{x})$ is sharp, if it is a small value close to 0 for a true negative case, or a big value close to 1 for a true positive case. The more confident the predicted event probabilities are, the smaller the Brier score is. A small Brier score indicates that the prediction model is sharp.

A random guessing classifier assigns a $\hat{p}(\boldsymbol{x})$ of 0.5 to every observation, resulting in a Brier score of 0.25. A Brier score of 0 shows perfect sharpness. Meanwhile, a Brier

score of 1 means perfect non-sharpness, where the model assigns $\hat{p}(\boldsymbol{x}) = 1$ to all the true negatives and $\hat{p}(\boldsymbol{x}) = 0$ to all the true positives. However, these two situations will never happen in practice. But hopefully, the $\hat{p}(\boldsymbol{x})$ is rather close to 0 for a true negative case, or close to 1 for a true positive case. In that case, a rather small Brier score will be obtained, and the prediction model is then sharp.

However, for class imbalanced data with a majority of negative cases, a very small Brier score does not necessarily mean that a classifier is sharp and distinguishes well between the negative cases and the positive cases. The Brier score has limitations when applied to very rare events forecasts (Benedetti, 2010). Assume, for instance, that a classifier gives $\hat{p}(\boldsymbol{x}) \approx 0$ to every observation in a test set of 990 negative cases and 10 positive cases. All the $\hat{p}(\boldsymbol{x})s$ are approximately 0 for the 990 true negatives, which is acceptable. The Brier score is only 0.01, which is very small. This classifier seems to show good performance in terms of the Brier score, but all the true positives are assigned very small (close to 0) $\hat{p}(\boldsymbol{x})s$. In this case, we cannot say that the prediction model is sharp, even though the Brier score is rather small. The Brier score behaves like the accuracy and the AUC, that they are biased towards the majority class when dealing with heavily class imbalanced data, and may give overly optimistic prediction results, despite the fact that they are useful measures in class balanced learning scenarios, where both the positive class and the negative class are equally important.

To further examine how much the Brier score of each class contributes to the total Brier score, we give respectively the Brier score bs_1 of the positive class and the Brier score bs_0 of the negative class by

$$bs_1 = \frac{1}{n_1} \sum_{i=1}^{n_1} (\hat{p}(\boldsymbol{x_i}) - 1)^2,$$

and

$$bs_0 = \frac{1}{n_0} \sum_{i=1}^{n_0} (\hat{p}(\boldsymbol{x_i}))^2.$$

Here, n_1 and n_0 denote respectively the number of the true positives and the number of the true negatives.

Chapter 3

Simulation study

This chapter is devoted to a simulation study, where we generate synthetic data sets and use the penalized logistic regression methods for fraud detection. Different measures of predictive performance are then explored, in particular, in the search for the optimal values of the tuning parameters.

3.1 Data simulation methods

The simulated data should represent the important features of the real data within fraud detection, that they are class imbalanced and high-dimensional. This section introduces the statistical methods and algorithms used to generate class imbalanced and high-dimensional data.

3.1.1 Generating covariates from copulas

In fraud detection, the data one encounters typically consist of a mixture of discrete and continuous covariates. Besides, there exists dependence between certain covariates. To tackle a situation like this, we need to find a way to generate such data, and this may be done by using a copula. The Gaussian copula is used in this thesis to simulate dependent covariates with arbitrary marginal distributions, which is achieved by applying the inverse transformation method. The Gaussian copula with a parameter matrix $\boldsymbol{\rho}$ describes the dependence structure between the covariates.

Copulas

A multivariate copula $C_{U_1,\ldots,U_p}(u_1,\ldots,u_p) = P(U_1 \leq u_1,\ldots,U_p \leq u_p)$ is a joint distribution function for a uniform and dependent vector $(U_1,\ldots,U_p)$. The variables $U_1,\ldots,U_p$ follow the uniform distribution on the interval $[0,1]$, and they can be designed

to be dependent on each other through their joint function $C_{U_1,\ldots,U_p}(u_1,\ldots,u_p)$. For that reason, copulas can be used to describe and to model the dependence structure of random variables. Sklar's theorem(1959) provides the theoretical foundation for the application of copulas. It states basically that any multivariate joint distribution function of a set of random variables might be expressed in terms of its univariate marginal distributions and a copula C. Suppose that p continuous random variables $X_1,\ldots,X_p$ have a joint cumulative distribution function $F_{X_1,\ldots,X_p}(x_1,\ldots,x_p) = P(X_1 \leq x_1,\ldots,X_p \leq x_p)$, and their marginal cumulative distribution functions are given by $F_{X_1}(x_1),\ldots,F_{X_p}(x_p)$. As a result of the probability integral transform, the marginal cumulative distribution functions $F_{X_j}(X_j)$, for $j = 1,\ldots,p$, are uniformly distributed on the interval $[0,1]$. It follows from the Sklar's theorem(1959) that

$$F_{X_1,\ldots,X_p}(x_1,\ldots,x_p) = C_{U_1,\ldots,U_p}(u_1,\ldots,u_p),$$

where $u_1 = F_{X_1}(x_1),\ldots,u_p = F_{X_p}(x_p)$. The copula $C_{U_1,\ldots,U_p}(u_1,\ldots,u_p)$ is a multivariate joint distribution function of variables $U_1,\ldots,U_p$ and is unique. It contains all information about the dependence structure between random variables $X_1,\ldots,X_p$. Combined with the marginal distributions $F_{X_1},\ldots,F_{X_p}$, it gives the joint distribution $F_{X_1,\ldots,X_p}(x_1,\ldots,x_p)$. Sklar's theorem is also valid when at least one of the X_is is discrete, but $F_{X_j}(X_j)$ is no longer uniform and the copula is not unique.

The Gaussian copula

The Gaussian copula, which allows for an elliptical dependence structure, is chosen for simulating correlated marginals of covariates. Suppose that $\Phi_{\boldsymbol{\rho}}(z_1,\ldots,z_p)$ is the joint cumulative distribution function of a multivariate normal distribution over $\mathbb{R}^p$, with a mean vector $\mathbf{0}$ and a covariance matrix equal to a correlation matrix $\boldsymbol{\rho}$. The Gaussian copula with a parameter matrix $\boldsymbol{\rho}$ is givne by

$$C^{Gauss}_{U_1,\ldots,U_p;\boldsymbol{\rho}}(u_1,\ldots,u_p) = \Phi_{\boldsymbol{\rho}}(\Phi^{-1}(u_1),\ldots,\Phi^{-1}(u_1)),$$

where $\boldsymbol{\rho}$ is a correlation matrix, and $\Phi^{-1}(\cdot)$ is the inverse cumulative distribution function of the standard univariate normal distribution. A possible way for generating samples $u_1,\ldots,u_p$ from the Gaussian copula is to first sample $z_1,\ldots,z_p$ from a p-dimensional multivariate normal distribution with means of 0, unit variances and a correlation matrix $\boldsymbol{\rho}$. Then we apply the cumulative distribution function of the standard normal distribution $\Phi(\cdot)$ to these samples $z_1,\ldots,z_p$, and obtain uniform values $u_1 = \Phi(z_1),\ldots,u_p = \Phi(z_p)$. The joint distribution function of the corresponding variables $U_1,\ldots,U_p$ is the Gaussian copula.

Sampling by inversion

Since fraud data usually have both continuous and discrete covariates, and they generally follow different marginal distributions, we use the inverse transform method to generate such data.

Suppose that a continuous variable X has a distribution function $F_X(x)$ which is strictly increasing with inverse $x = F^{-1}(u)$. Then $X = F^{-1}(U)$, where U is uniformly distributed on the interval $[0, 1]$, and it follows the distribution $F_X(x)$. Consequently, we have a general sampling technique for generating continuous covariates. That is to say, in the setting of p dimensions, we may apply the inverse distribution functions $F_1^{-1}(\cdot), \ldots, F_p^{-1}(\cdot)$ to the uniform variables $U_1, \ldots, U_p$, so as to generate variables $X_1, \ldots, X_p$ that follow respectively the desired marginal distributions $F_{X_1}, \ldots, F_{X_p}$.

In this simulation study, we consider only the Bernoulli distribution for discrete covariates. Suppose that we want to generate a discrete variable X which follows the Bernoulli distribution with probability of success p_B(i.e., $X \sim Bernoulli(p_B)$), namely, $P(X = 1) = p_B$ and $P(X = 0) = 1 - p_B$, for some $p_B \in (0, 1)$. Then a sample x from this distribution is obtained as: let u be a randomly generated value of the uniformly distributed variable U on the interval $[0, 1]$, then $x = 1$, if $u > 1 - p_B$, and $x = 0$, otherwise.

Through the Gaussian copula, the whole procedure for generating values of covariates for one observation is given in Algorithm 3.1.1.

Algorithm 3.1.1 Sampling the covariates from the Gaussian copula

1: Draw $\boldsymbol{z} = (z_1, \ldots, z_p) \sim N(\mathbf{0}_p, \boldsymbol{\rho})$ ▷ $\boldsymbol{\rho}$ is a $p \times p$ correlation matrix.
2: Generate $u_j = \Phi(z_j)$, $j = 1, \ldots, p$
3: **for** $j = 1, \ldots, p$ **do**
4: **if** $X_j \sim Bernoulli(p_B)$ **then**
5: $$x_j = \begin{cases} 1, & u_j > 1 - p_B \\ 0, & \text{otherwise} \end{cases}$$
6: **else**
7: $x_j = F_j^{-1}(u_j)$
8: **end if**
9: **end for**
10: **Return**: $x_1, \ldots, x_p$

The correlation matrix $\boldsymbol{\rho}$

The correlation matrix $\boldsymbol{\rho}$ of the underlying multivariate normal distribution $\Phi_{\boldsymbol{\rho}}(\cdot)$ needs to be defined, before sampling the covariates from the Gaussian copula as presented in Algorithm 3.1.1. After applying the inverse transform method to uniform values $u_1, \ldots, u_p$, the correlation matrix for the finally generated samples $x_1, \ldots, x_p$ will not end up the same as the correlation matrix $\boldsymbol{\rho}$ of the underlying function $\Phi_{\boldsymbol{\rho}}(\cdot)$. However, the Gaussian copula function with parameter matrix $\boldsymbol{\rho}$ contains all the dependence information between random variables, independently of their marginal distributions. Hence, the dependence structure among random variables, which is represented by the copula function, is not altered. For p covariates, we simply sample a valid $p \times p$ correlation matrix $\boldsymbol{\rho}$ randomly, using the R function **randcorr**.

3.1.2 Generating a data set

To simulate class imbalanced data, we may directly work with the proportion τ of fraudulent cases in the data, in other words, we prefix and adjust the value of τ to achieve the desired level of class imbalance in the data. A possible way of sampling such class imbalanced data through the prefixed τ is inspired by Bayes' theorem for classification.

Bayes' theorem

According to Bayes' theorem, the event probability $P(Y = 1|\boldsymbol{X})$ may be given by

$$\begin{aligned} P(Y = 1|\boldsymbol{X}) &= \frac{P(\boldsymbol{X}|Y = 1)\tau}{P(\boldsymbol{X})} \\ &= \frac{P(\boldsymbol{X}|Y = 1)\tau}{P(\boldsymbol{X}|Y = 1)\tau + P(\boldsymbol{X}|Y = 0)(1 - \tau)}, \end{aligned} \tag{3.1}$$

where $\tau = P(Y = 1)$ and $1 - \tau = P(Y = 0)$. A rather small value of τ, for instance, $\tau = 0.01$, means that the data are highly class imbalanced.

This inspires an adequate way of sampling class imbalanced data. With a prefixed value of τ, we could generate the outcomes for both the positive and the negative class, separately. Meanwhile, their covariates may be generated from the distributions $P(\boldsymbol{X}|Y = 1)$ and $P(\boldsymbol{X}|Y = 0)$, respectively, as explained below. Consequently, the true event probabilities of the simulated data can be computed using the Equation (3.1).

Data generation design

In what follows, we outline in detail the whole procedure of generating class imbalanced data. Suppose we want to generate an $n \times p$ data set with a proportion τ of positive

cases. For $i = 1, \ldots, n$, let the outcome Y_i have a Bernoulli distribution with probability of success τ, i.e., $Y_i \sim Bernoulli(\tau)$. As a result, there will be approximately $n \cdot \tau$ of positive cases and $n \cdot (1 - \tau)$ of negative cases. It is possible to simulate data sets with different ratios of class imbalance with $\tau \leq 0.2$, since fraud data are usually very class imbalanced.

In addition, the correlation matrices for both classes need to be determined before the sampling of the covariates. The easiest way is to first randomly generate a positive definite correlation matrix for each class, and then set only the correlations between the significant covariates to be non-zero. Only 10% of the covariates are set to be significant or truly influential for the outcome (see Subsection 3.2.2). We generate data from a sparse model in the sense that only a few of the covariates affect the outcome. As a result, it is easier to obtain resulting correlation matrices that are positive definite. We denote the correlation matrix of the negative class by $\boldsymbol{\rho_0}$ and the correlation matrix of the positive class by $\boldsymbol{\rho_1}$. We use different correlation matrices $\boldsymbol{\rho_0}$ and $\boldsymbol{\rho_1}$, in order that the distributions of covariates in the two classes, i.e., $P(\boldsymbol{X}|Y = 1)$ and $P(\boldsymbol{X}|Y = 0)$, do not overlap too much. Otherwise, it might be too challenging for the classifier to separate between the positive and negative class.

If the generated outcome $y = 0$, that is, the case is negative, we sample the covariate values $\boldsymbol{x} = (x_1, \ldots, x_p)$ from the Gaussian copula with the parameter matrix $\boldsymbol{\rho_0}$, following the procedure in Algorithm 3.1.1. However, if the generated outcome $y = 1$, we sample the covariate values $\boldsymbol{x} = (x_1, \ldots, x_p)$ from the Gaussian copula with the parameter matrix $\boldsymbol{\rho_1}$, following the procedure in Algorithm 3.1.1 as well.

Further, the marginal distribution of each covariate X_j is of the same distribution family, e.g., the normal distribution, in both classes, for $j = 1, \ldots, p$. The covariates have the same parameter values of marginals in both classes, except for the truly influential covariates, that have different parameter values of marginals in the two classes. While the parameter values of marginals in the negative class are kept fixed, we adjust the parameter values for some covariates' marginals in the positive class. The modification should be moderate, in order that the classifier's prediction performance is hopefully just good enough for class balanced data. Otherwise, too much, or too little change in the parameters of positives' marginals will make it too easy, or too demanding for the classifiers to detect well the positives well with highly class imbalanced data. Either case will make the whole simulation study lose meaning and usefulness for real fraud detection problems. The details about the marginal distributions of both classes are presented in Section 3.2.1.

Lastly, an $n \times p$ data set is obtained by putting together the positive and negative cases with corresponding covariate values. The whole procedure for generating an $n \times p$ data

set with a given ratio τ of class imbalance is given in Algorithm 3.1.2.

Algorithm 3.1.2 Generating a data set with a given ratio of class imbalance

Input: $n, p, \tau, \boldsymbol{\rho_0}, \boldsymbol{\rho_1}$ ▷ τ is the prefixed ratio of class imbalance.

1: **for** $i = 1, \ldots, n$ **do**
2: Draw $y_i \sim Bernoulli(\tau)$
3: **if** $y_i = 0$ **then**
4: Draw $\boldsymbol{x_i}$ from $F(\boldsymbol{X}|Y = 0)$ #Algorithm 3.1.1
5: **else if** $y_i = 1$ **then**
6: Draw $\boldsymbol{x_i}$ from $F(\boldsymbol{X}|Y = 1)$ #Algorithm 3.1.1
7: **end if**
8: **end for**
9: **Return**: $(y_i, \boldsymbol{x_i})$, $i = 1, \ldots, n$

As we can see, at step 4 and step 6, $\boldsymbol{X}$ here is the vector of all p covariates with $\boldsymbol{X} = (X_1, \ldots, X_p)$, and $F(\cdot|Y = y)$ is the copula with correlation matrix $\boldsymbol{\rho}_y$ and marginal distributions $F(X_j|Y = y)$, for $y = 0, 1$, and $j = 1, \ldots, p$.

3.2 Simulation design

This section presents the specific settings which the simulations are run with, including the information about experiment design, specified values of the parameters of marginals in each class, the implemented cross-validation method, the implementation of regularization methods in R, model optimization criteria and supplementary evaluation metrics.

3.2.1 Experiment design

Data sets with various ratios of class imbalance and sample sizes are simulated, in order to explore different simulation settings with respect to the objective of the simulation study. To manipulate the level of class imbalance, we use the method discussed in Subsection 3.1.2. The ratio τ of class imbalance is set to be 0.01, 0.05, 0.1, or 0.2. Each number represents a rather high level of class imbalance. The number p of covariates in all the data sets is set to 1000. Data sets with different dimensionalities will be obtained by altering the sample size n, which is set to be 100, 800, 1000, or 3000. The ratio $\frac{n}{p}$ is then given by 0.1, 0.8, 1 and 3, respectively. As a result, it leads to 16 different types of training sets in total.

Since, in the simulation study, it is possible to generate an independent test set separately from the training set, it is practical to have a big test set with many observations, in

order that the test set is representative of the underlying data, hence, the performance of classifiers evaluated on the test set will not be affected by data rarity. For each $n \times 1000$ training set with a ratio τ of class imbalance, an independent 3000×1000 test set is sampled with the same ratio of class imbalance. Every learning algorithm is applied to 100 pairs of such train-test data sets. The results obtained over all the test sets are averaged in the end. The average results of 100 simulations are reported in Section 3.3.

3.2.2 Model for simulated data

Before generating a data set with the method presented in Subsection 3.1.2, the correlation matrices for both classes and the marginal distributions $F(X_j|Y = y)$, for $y = 0, 1$, and $j = 1, \ldots, p$, should be specified. We choose marginal distributions for the covariates, including the normal distribution, the gamma distribution, and two Bernoulli distributions with different probabilities of success. Therefore, in both classes, the covariate X_j comes from a normal distribution, for $j = 1, \ldots, 300$, a gamma distribution, for $j = 301, \ldots, 600$, a Bernoulli distribution, for $j = 601, \ldots, 800$, and another Bernoulli distribution, for $j = 801, \ldots, 1000$.

Only the first 10% of covariates within each distribution family are considered to be truly influential for the outcome. Therefore, only correlations between these relevant covariates are set to be non-zero, and it is done on two randomly generated positive definite correlation matrices. The resulting correlation matrices will be used for generating all the data sets in the simulation study.

For simplicity, the marginal distributions of the negative class are given by

$$\begin{gathered} X_j|Y = 0 \sim N(0, 1),\ for\ j = 1, \ldots, 300, \\ X_j|Y = 0 \sim Gamma(2, 2),\ for\ j = 301, \ldots, 600, \\ X_j|Y = 0 \sim Bernoulli(0.1),\ for\ j = 601, \ldots, 800, \\ \text{and } X_j|Y = 0 \sim Bernoulli(0.5),\ for\ j = 801, \ldots, 1000. \end{gathered} \tag{*}$$

Each covariate comes from the same distribution family in both classes. For the first 10% of covariates within each distribution family, different values of parameters are used for the positive class. Accordingly, the remaining 90% of covariates within each distribution family of the positive class, should have exactly the same marginal distributions as in the negative class. This ensures that only the first 10% of the covariates within each distribution family have an effect on the outcome. The marginal distributions of the positive class are given by

$$X_j|Y = 1 \sim N(0.6, 1.5),\ for\ j = 1, \ldots, 30,$$

$$
\begin{aligned}
X_j|Y=1 &\sim Gamma(2.05, 2.01),\ for\ j = 301, \ldots, 330, \\
X_j|Y=1 &\sim Bernoulli(0.2),\ for\ j = 601, \ldots, 620, \qquad (**) \\
X_j|Y=1 &\sim Bernoulli(0.4),\ for\ j = 801, \ldots, 820,
\end{aligned}
$$

and for the remaining indices, it is as in (*).

The procedure for deciding the above parameters is illustrated as below. We first generate a balanced 3000×1000 training set with approximately half positive cases and half negative cases, where the marginal distributions of the negative class are prefixed as in (*). The positive cases are simulated with freely chosen parameter values of the first 10% of covariates within each distribution family. Then, we fit the ridge logistic regression with respect to the AUC in ten-fold cross-validation of the penalty parameter λ. At last, the fitted model inserted with the optimal value of λ^{opt} is evaluated on an independent balanced 3000x1000 test set in terms of the AUC. This procedure is repeated 10 times, and all the obtained AUC values are averaged into one final result. This final AUC should be just good enough, as explained in Section 3.1.2. The Hosmer Jr et al. (2013) gives the general rules for determining whether an AUC describes good discrimination. After testing with different parameter values of the first 10% of covariates within each distribution family of the positive class, we achieved an outstanding discrimination with an AUC slightly larger than 0.9, by setting the marginal distributions for the positive class as in (**).

3.2.3 Implemented cross-validation

Stratified cross-validation is used in the simulation study, mainly as a consequence of class imbalance. It is challenging to carry out stratified ten-fold (i.e., $k = 10$) cross-validation on all 16 types of training sets, as presented in Section 3.2.1, since some types of training sets may have less than ten positives. Although the expected number of positives in the training set, i.e., $\tau \cdot n$, equals ten, it is still possible that there are less than ten positives in the actual sampled training set, which may make the training very difficult. For this reason, stratified ten-fold cross-validation is applied only when the expected number of positives in a training set is larger than ten. When the expected number of positives in a training set is larger than five and less than ten, stratified five-fold (i.e., $k = 5$) cross-validation is used. For those training sets with no more than five expected number of positives, no experiments are conducted. Therefore, it leaves us 14 types of simulated training sets, since it can very difficult to fit models on a 100×1000 training set with $\tau = 0.01$, or 0.05.

3.2.4 The logistic regression without penalty

Logistic regression is often plagued with degeneracies when $p > n$, and exhibits wild behavior even when n is close to p (Hastie et al., 2021). When handling high-dimensional data, it is often easier for the logistic regression model to find a possible set of parameters, such that the positive and negative class in the training set may be perfectly separated. This full model, which is fitted without constraints on the parameters, has a tendency to overfit the data, and it may not generalize well. But still, we fit the logistic regression model by the R function **glm**, even though the model may not converge occasionally, and report its performance for $n = 3000$.

3.2.5 Implementation of the regularization methods

The ridge and lasso penalty

The R function **cv.glmnet** is used for performing cross-validation on the training set, and it automatically produces the optimal value of the shrinkage parameter λ for penalized logistic regression models. With argument $alpha = 0$, the ridge regression is called, while the lasso regression is called for argument $alpha = 1$. As for the argument $foldid$, it may take the user-supplied folds (Hastie et al., 2021). Here, the labels are generated with stratified cross-validation in Algorithm 2.1.1, since the simulated data are highly class imbalanced. The default value of argument *nlambda* is 100 in the R function **cv.glmnet**, such that 100 λ values are automatically constructed by the program, and the model fit is computed for the *lambda* sequence (Hastie et al., 2021).

The λ value that gives the minimum mean cross-validated loss is chosen, for both the ridge and the lasso regression. Since selecting the most parsimonious model with low error is not the goal, we do not consider the value of λ obtained from one stand error rule, where the most parsimonious or regularized model within one standard error of the minimum is picked (Hastie et al., 2009). Besides, with high-dimensional data, the one standard error rule may suit the lasso and elastic net better where some parameters may be actually shrunk to zero. Using the one standard error rule means that the interpretability is preferred over the predictive ability of a model, which is not the case in this thesis. Here, we are interested in detecting the rare cases.

The elastic net penalty

For the elastic net penalty, a grid of values of the shrinkage parameters α and λ, are screened, in order to find the optimal regularization path, namely, the combination of values of α and λ that produces the minimum average cross-validated loss. We use a sequence of α values, i.e., $0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8$ and 0.9. For each α value, an automatically constructed sequence of 100 λ values is used for computing the model fit.

3.2.6 Optimality measures used in the cross-validation of the penalty parameters

The classification models should be trained under an appropriate loss function, or an appropriate measure of prediction performance (Daskalaki et al., 2006). In model selection, the class imbalanced training set is divided further into training and validation folds, by stratified cross-validation (see Subsection 2.5.2), and a certain performance measure is incorporated into the penalized logistic regression model when searching for the optimal values of penalty parameters. A model fitted to the training folds is evaluated on the validation fold with respect to this performance measure.

The deviance and the AUC are available as optimality criteria for the cross-validation by specifying *type.measure* in the *R* function **cv.glmnet**. To minimize the misclassification error is to maximize the overall prediction accuracy, which is not considered in the simulation study. It is due to the fact that the overall prediction accuracy is biased towards the majority class, and does not distinguish between the numbers of correctly detected cases of different classes (Galar et al., 2012), let alone it fails the main interest of correctly detecting the fraudulent cases within fraud detection. Overall, it may result in inaccurate conclusions, and it is not an adequate measure for class imbalanced data. Since the performance measure AUCPR is not available in built-in optimality measures of the *R* function **cv.glmnet**, it is necessary to build our own function for using AUCPR as the optimality measure for cross-validation.

3.2.7 More evaluation metrics

In the experiments, the logistic regression model with each of the ridge, lasso and elastic net penalties are optimized with respect to each of the three types of optimality measures, i.e., the deviance, the AUC and the AUCPR, respectively. Hence, it leads to nine models for each simulated training set. The prediction performance of these fitted models is evaluated and compared using the following evaluation metrics: the AUCPR, the AUC, the maximal AGm with corresponding TPR and TNR, the maximal Gm with corresponding TPR and TNR, the Brier score, the bias, the Rmse and the actual ranking score.

All the mentioned evaluation metrics are discussed in Section 2.6, except the last three measures, which involve the true event probabilities (i.e., $p(\boldsymbol{x})s$), which are unknown in real applications. The true event probabilities can be computed for simulated data sets, as briefly explained in Subsection 3.1.2.

Bias and Rmse

The bias and Rmse are respectively given by

$$Bias = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{p}(\boldsymbol{x_i}) - p(\boldsymbol{x_i})\right),$$

and

$$Rmse = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{p}(\boldsymbol{x_i}) - p(\boldsymbol{x_i})\right)^2}.$$

Both the bias and the Rmse measure the difference between the predicted event probabilities and the true event probabilities. Unlike the Rmse, the bias is not always non-negative. The sign of the bias may be negative or positive, meaning that the model systematically underestimates, or overestimates the true event probabilities, respectively.

The Rmse is a measure of the spread, which is always non-negative. The prediction error in an observation is the difference between its predicted event probability and its true event probability. The Rmse is capable of reflecting some large errors, since it gives more weight to errors with large absolute values rather than errors with small absolute values. This can also be seen as a weakness, as in situations where most of the errors are rather small, a few large errors (e.g., outliers) can increase RMSE a lot. Nevertheless, the Rmse has a decent discrimination ability among several models, and it usually reveals well the difference between the performance of models (Chai and Draxler, 2014).

To further examine how much a model systematically underestimates or overestimates the predicted event probabilities in each class, we give respectively bias b_1 of the positive class and bias b_0 of the negative class by

$$b_1 = \frac{1}{n_1}\sum_{i=1}^{n_1}\left(\hat{p}(\boldsymbol{x_i}) - p(\boldsymbol{x_i})\right),$$

and

$$b_0 = \frac{1}{n_0}\sum_{i=1}^{n_0}\left(\hat{p}(\boldsymbol{x_i}) - p(\boldsymbol{x_i})\right).$$

Here, n_1 and n_0 denote respectively the number of the true positives and the number of the true negatives.

The actual ranking ability

The AUC examines a model's ranking ability in giving higher predicted scores to the true positives over the true negatives, so it measures a model's overall ranking ability over all cases. In fraud detection, it is practical and tempting to investigate only a few cases that are most likely to be fraudulent according to the predicted event probabilities. Otherwise, it can be too time consuming and expensive, or most likely impossible to undertake thorough investigations over all cases. To reflect that, another measure that evaluates the model's actual ranking ability among a few top-ranked cases, is also proposed.

We investigate the actual ranking ability by looking into a few top-ranked cases according to the predicted event probabilities, of which we check how many are among the top-ranked cases according to the true event probabilities. Here, let n_1 be the number of the true positives in the test set of 3000 observations, and only the top n_1 cases with the highest predicted event probabilities are examined. The actual ranking score is given by

$$actual\,ranking\,score = \sum_{i=1}^{n_1} \mathbb{1}_I\left\{\hat{I}_i\right\}, \tag{3.2}$$

where the indicator function

$$\mathbb{1}_I\left\{\hat{I}_i\right\} = \begin{cases} 1, \ if & \hat{I}_i \in I, \\ 0, & \text{otherwise.} \end{cases}$$

Here, $\hat{I} = \left\{i : \hat{p}(\boldsymbol{x_i}) \geq \hat{P}_{(n_1)}\right\}$, and $I = \left\{i : p(\boldsymbol{x_i}) \geq P_{(n_1)}\right\}$. The $\hat{I}_i$ is the ith element in set $\hat{I}$. The $\hat{P}_{(n_1)}$ and $P_{(n_1)}$ denote respectively the n_1th highest predicted event probability and the n_1th highest true event probability. The set $\hat{I}$ consists of the indices of the cases that are among the top n_1 cases with the highest predicted event probabilities. The set I consists of the indices of the cases that are among the top n_1 cases with the highest true event probabilities.

For any of the top-ranked n_1 cases according to the predicted event probabilities, we do not consider whether or not it has a predicted event probability that is close to its true event probability, as long as it is among the top-ranked n_1 cases according to the true event probabilities. And we do not consider the order among those cases either. The actual ranking score investigates the proportion of true positives among the cases that are most likely to be positive according to the predicted event probabilities.

3.2.8 Parallel computing

The stratified cross-validation for methods optimized with respect to the deviance and AUC, are run in parallel, since the *R* function **cv.glmnet** supports parallel computing, which can substantially speed up the computation process, especially for large-scale problems (Hastie et al., 2021). It does save computation time of the simulation study to some degree. Unfortunately, no parallel computing is performed in stratified cross-validation for the regularization models which are optimized with respect to the AUCPR. Therefore, compared to other optimality measures, it takes longer to find the optimal values of the tuning parameters that give the highest cross-validated AUCPR.

3.3 Results

This section presents the experimental results of the simulation study. In every table that is shown in this section, the deviance, the AUC and the AUCPR are measures used in cross-validation of the penalty parameters. The ratio τ of class imbalance is the proportion of positive samples. The sample size n is the number of observations in the training set. Elnet stands for the elastic net method. The standard logistic regression model, without any constraints imposed on the coefficients, is denoted by LR. Its results are reported for $n = 3000$, which is merely used as a reference, and is not included in finding out the highest value in each simulation setting.

3.3.1 Ranking ability

AUCPR or AUC

Both the AUC and the AUCPR can be used to evaluate a model's ranking ability, but they may rank the models in different orders, and give different results of the same model in the same simulation setting, especially for class imbalanced data. The purpose here is to decide whether they are appropriate measures of the overall ranking ability, for model selection in the setting of fraud detection.

Table 3.1 presents the average AUCPR (*upper*) and AUC (*bottom*), obtained with LR and three penalized logistic regression models optimized with respect to each of the three model optimization criteria, in each of the fourteen simulation settings. The highest value in each simulation setting is printed in bold. Further, we denote, for instance, the lasso regression optimized with respect to the AUCPR by $lasso^{AUCPR}$.

The overall ranking ability of every regularization method, evaluated either by the AUCPR or the AUC, is improved when either the proportion τ of positives, or the sample size n is increased, that is, when there is less class imbalance in the data, or

Table 3.1: Average AUCPR (*upper*) and average AUC (*bottom*). The highest value in each simulation setting is printed in bold. If a model has a score with an asterisk, it performs worse than *elastic net*^{AUCPR} at a significance level of 0.05.

			Ridge			Lasso			Elnet		
τ	n	LR	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR
0.01	800	-	0.08*	0.08*	0.08*	0.12	0.12	0.12	**0.15**	**0.15**	0.14
	1000	-	0.10*	0.10*	0.10*	0.14	0.15	0.14	**0.16**	0.14	**0.16**
	3000	0.11	0.28*	0.26*	0.29*	0.40*	0.38*	0.40*	0.42	0.43	**0.44**
0.05	800	-	0.54*	0.55*	0.55*	0.62*	0.62*	0.61*	0.65*	0.67	**0.69**
	1000	-	0.60*	0.60*	0.60*	0.67*	0.67*	0.66*	0.68*	0.70*	**0.73**
	3000	0.48	0.73*	0.78*	0.78*	0.82*	0.82*	0.82*	0.82*	0.83*	**0.85**
0.1	100	-	0.32*	0.32*	0.32*	0.31*	0.33*	0.32*	**0.38**	0.36	0.37
	800	-	0.75*	0.76*	0.76*	0.80*	0.80*	0.80*	0.81*	0.82*	**0.85**
	1000	-	0.79*	0.79*	0.79*	0.83*	0.82*	0.83*	0.83*	0.84*	**0.87**
	3000	0.63	0.84*	0.88*	0.89*	0.90*	0.90*	0.90*	0.90*	0.90*	**0.91**
0.2	100	-	0.56*	0.56*	0.56*	0.56*	0.55*	0.53*	**0.62**	**0.62**	0.60
	800	-	0.88*	0.88*	0.88*	0.91*	0.91*	0.91*	0.91*	0.92*	**0.93**
	1000	-	0.90*	0.90*	0.90*	0.91*	0.92*	0.92*	0.91*	0.92*	**0.94**
	3000	0.75	0.92*	0.94*	0.94*	0.94*	**0.95***	**0.95***	0.94*	**0.95***	**0.95**
0.01	800	-	**0.78**	**0.78**	**0.78**	0.72*	0.73*	0.73*	**0.78**	0.77	0.77
	1000	-	**0.80**	**0.80**	**0.80**	0.74*	0.75*	0.76*	0.79	0.78	0.79
	3000	0.81	0.89*	0.86*	0.90*	0.90*	0.89*	0.90*	**0.91**	**0.91**	**0.91**
0.05	800	-	0.90*	0.91*	0.91*	0.91*	0.90*	0.90*	0.92*	0.92	**0.93**
	1000	-	0.92*	0.92*	0.92*	0.92*	0.92*	0.92*	0.93*	0.93*	**0.94**
	3000	0.88	0.95*	0.96*	0.96*	**0.97***	**0.97***	**0.97***	**0.97***	**0.97***	**0.97**
0.1	100	-	**0.76**	**0.76**	**0.76**	0.69*	0.70*	0.70*	**0.76**	0.73	0.75
	800	-	0.94*	0.94*	0.94*	0.94*	0.94*	0.94*	0.95*	0.95*	**0.96**
	1000	-	0.95*	0.95*	0.95*	0.95*	0.95*	0.95*	**0.96***	**0.96***	**0.96**
	3000	0.89	0.96*	0.97*	0.97*	0.97*	0.97*	0.97*	0.97*	0.97*	**0.98**
0.2	100	-	0.80	**0.81**	**0.81**	0.77*	0.76*	0.74*	**0.81***	**0.81**	0.79
	800	-	0.95*	0.95*	0.95*	0.96*	0.96*	0.96*	0.96*	0.96*	**0.97**
	1000	-	0.96*	0.96*	0.96*	0.96*	0.96*	**0.97***	0.96*	**0.97***	**0.97**
	3000	0.90	0.97*	**0.98***	**0.98***	**0.98***	**0.98***	**0.98***	**0.98***	**0.98***	**0.98**

more observations to use in the training process. The improvement is more distinct with respect to the AUCPR. Compared to the AUC, the AUCPR is more sensitive towards both the changes in the ratio τ of class imbalance, and the changes in the ratio between the sample size n and the number p of covariates. In particular, the AUCPR increases drastically as τ increases from 0.1 to 0.2, whereas the AUC remains almost unaltered.

The AUCPR is also better at selecting models, since it can distinctly reveal the performance differences that the AUC cannot. We observe in Table 3.1 that the ridge and elastic net regression produce similar AUCs in many simulation settings. It may be wrongly concluded that they have similar performance, if no significance tests are conducted. Meanwhile, the elastic net often wins substantially over the ridge in terms of the AUCPR. As a matter of fact, the *elastic net*^{AUCPR} outperforms the ridge regression

at a significance level of 0.05 in almost all simulation settings, using a one-sided Wilcoxon signed-rank test and the Holm's adjustment method for multiple tests afterwards.

The AUCPR is very informative, and actually reveals rather poor performance of all methods for severe class imbalance (e.g., $\tau = 0.01$). Meanwhile, the AUC, regardless of the ratio τ of imbalance, often gives excessively high values around 0.8, which is considered at the low end of excellent discrimination (Hosmer Jr et al., 2013). The AUC indeed provides an overly optimistic view of the performance in imbalanced classification scenarios, which may result in misleading conclusions of the models' performance.

As discussed in Chapter 2, the AUCPR considers the precision, instead of the TNR, so it reveals the accurate prediction performance of a model with the focus on the positive class in imbalanced learning, whereas the AUC is biased towards the negative cases in majority. Based on the experimental results and the aforementioned advantages of the AUCPR, the AUCPR is more appropriate than the AUC in this case. Here, the AUCPR should be preferred as the measure of prediction performance.

Does the regularization work?

Consequently, we will mainly look at the results of the AUCPR when comparing the differences between the models' performance. For high-dimensional data ($n \leq p$ in this thesis), all regularization methods perform much better than a random guessing classifier, whose expected AUCPR is determined by the ratio τ of class imbalance. The difference between the prediction performance of the regularization methods and LR is considerable, except for $\tau = 0.01$. A rather small proportion of positives in the training data is expected to make it much more difficult for all the regularization methods to learn the difference between the two classes.

For $n = 3000$, LR actually has some predictive ability and performs better than a random guessing classifier. All regularization methods perform better, in terms of the AUCPR, than LR at a significance level of 0.05. The ridge regression shrinks the coefficients of correlated variables towards each other, which alleviates the problem that the coefficients of those correlated variables may be poorly determined and have high variance (Hastie et al., 2009). This probably explains why the ridge regression also performs better than LR.

On average the lasso performs better than the ridge. The model we simulated from is quite sparse for $n \neq 100$, which suits the lasso well. For the dense scenario $n = 100$, the ridge performs slightly better than, if not similarly to, the lasso. In general the elastic net performs better than the lasso, since the former can also handle correlated variables like the ridge.

Preferable measure for the cross-validation of the penalty parameters

We will investigate if the model optimization criteria affect the overall ranking ability of any of the regularization methods, and if there should be any preference for the measure used in cross-validation of the penalty parameters.

Both the ridge (in the high-dimensional setting) and the lasso are hardly affected by the optimality criteria, since each of them separately gives similar (if not identical) results in many simulation settings, regardless of the measure used in cross-validation of the penalty parameters. In the ridge regression, the penalty parameter λ chosen by the AUCPR is larger than the λ chosen by the AUC, which is again larger than the λ chosen by the deviance, as shown in Figure A.3 (in Appendix A.2). As the λ increases, all coefficients in the ridge are shrunk towards zero proportionally but will never be zero. Based on the structure of the sigmoid function, a larger shrinkage produces smaller predicted event probabilities for all observations, but will hardly change the order among them. So the λ over a wide interval gives approximately constant values of both the AUCPR and the AUC, as shown in Figure A.4 (in Appendix A.2). This is also observed in the lasso, where, in fact, different numbers of covariates are selected at different λ values. The reason why the lasso is insensitive to the changes in the λ is unknown. Nevertheless, which measure to use for selecting the optimal value of the λ does not really matter for the ridge and lasso regression.

For $n = 3000$ with $\tau = 0.05, 0.1$ and 0.2, both $ridge^{AUC}$ and $ridge^{AUCPR}$ perform better than $ridge^{deviance}$, but only with a small advantage. The reason is probably that the deviance usually chooses a much smaller λ than the AUC and the AUCPR. And in the aforementioned simulation settings, the deviance chooses a much too small λ.

The AUCPR should be preferred as the optimality criterion for the elastic net regression, since $elastic\ net^{AUCPR}$ consistently outperforms $elastic\ net^{deviance}$ and $elastic\ net^{AUC}$ at a significance level of 0.05, except for severe class imbalance and very high-dimensional feature, where they have similar performance. The AUCPR frequently chooses an α that is close to 0.1, as shown in Table A.1 (in Appendix A.2). Therefore, $elastic\ net^{AUCPR}$ is more close to the ridge ($\alpha = 0$) than the lasso ($\alpha = 1$), which may contribute to the fact that $elastic\ net^{AUCPR}$ often achieves the best prediction performance in the overall ranking ability.

If one considers also the interpretability of a model, the AUCPR should be preferred as the measure in cross-validation of the penalty parameters. For the lasso and elastic net regression, the AUCPR selects values of the penalty parameters that give almost consistently the highest proportion of truly influential covariates among the selected variables. This can be observed in Table 3.2, which presents the average proportion

of truly influential variables among the selected variables, in the lasso and elastic regression, respectively. Compared to *elastic net*^{deviance} and *elastic net*^{AUC}, the *elastic net*^{AUCPR} tends to select much fewer irrelevant covariates, but also a relatively high number of relevant covariates. A model probably has better prediction performance with much less noise from a big reduction of non-significant variables, considering that the true underlying data structure may be explained by only a few relevant covariates.

It is also of interest to investigate whether *elastic net*^{AUCPR} also outperforms the other regularization methods at all thresholds, provided that it actually performs the best in terms of the AUCPR. After all, the AUCPR is a summary statistic of the PR curve. The fact that a model outperforms the other in terms of the AUCPR, does not necessarily mean that the former is better than the latter at all thresholds. Simulations are run for $n = 1000$ with $\tau = 0.05$ and 0.1, respectively. Only the simulations where *elastic net*^{AUCPR} gives the highest AUCPR are shown. The PR curves of all regularization methods are plotted in the upper panel of Figure A.5 (in Appendix A.2). The ROC curves are given in the bottom panel as a reference. The PR curves of *elastic net*^{AUCPR} lie above those of other regularization models at almost all thresholds. This also applies to the ROC curve of *elastic net*^{AUCPR}, but the difference between the ROC curves is rather small. This behavior of *elastic net*^{AUCPR} in the PR space is actually observed in several other simulation settings.

Actual ranking score

Table 3.3 presents the average actual ranking scores of all methods. In general, the actual ranking score and the AUCPR in Table 3.1 follow a similar pattern. For instance, the ridge (in high-dimensional setting) and the lasso regression are barely affected by the measure used in cross-validation of the penalty parameters. For $n = 3000$, *ridge*^{deviance} performs the worst. The *elastic net*^{AUCPR} gives the best actual ranking score, when the data are neither severely class imbalanced nor very high-dimensional. Otherwise, there is not that much difference between the three optimality measures for the elastic net. Additionally, the *elastic net*^{AUCPR} outperforms other models at a significance level of 0.05, in many simulation settings.

The actual ranking ability of all methods improves as the data become less imbalanced, and for larger ratios between the sample size n and the number p of covariates. LR performs the worst. On average, the elastic net performs at least similarly to, if not better than the lasso, which again performs at least similarly to, if not better than the ridge. However, the difference between the performance of the lasso and elastic net regression is not that much for $n = 3000$.

Table 3.2: Average proportion of truly influential variables among the selected variables. The highest value is respectively printed in bold for the lasso and elastic net regression. The values in pair under the proportion, are respectively the average number of truly influential covariates that are selected, and the average number of total covariates that are selected.

		Lasso			Elnet		
τ	n	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR
0.01	800	**0.56**	0.43	0.55	0.33	0.18	**0.38**
		8 13	10 24	9 16	14 44	20 108	13 34
	1000	**0.56**	0.40	0.53	**0.32**	0.14	0.31
		11 20	14 34	12 24	18 57	34 234	19 62
	3000	0.51	0.50	**0.53**	**0.41**	0.27	0.38
		32 62	30 61	31 59	36 88	43 159	40 105
0.05	800	0.55	0.54	**0.67**	0.40	0.28	**0.50**
		33 61	32 60	28 42	39 99	44 157	38 76
	1000	0.50	0.49	**0.63**	0.37	0.39	**0.47**
		38 77	38 76	34 55	44 120	44 112	44 93
	3000	0.40	0.52	**0.58**	0.34	0.47	**0.51**
		59 147	55 106	54 93	62 180	57 121	62 121
0.1	100	**0.62**	0.52	0.56	0.29	0.20	**0.35**
		7 11	8 16	8 14	17 57	21 103	16 46
	800	0.47	0.50	**0.64**	0.34	0.34	**0.53**
		46 97	44 87	41 65	54 161	52 152	51 97
	1000	0.45	0.51	**0.62**	0.34	0.44	**0.51**
		50 111	47 92	45 72	56 165	52 118	55 109
	3000	0.38	0.57	**0.66**	0.35	0.57	**0.58**
		66 175	61 106	59 90	68 195	61 108	66 115
0.2	100	0.54	0.49	**0.68**	0.32	0.27	**0.49**
		13 24	13 28	9 14	24 73	27 99	16 34
	800	0.42	0.55	**0.64**	0.30	0.49	**0.56**
		54 130	50 91	48 76	61 205	54 111	58 104
	1000	0.40	0.56	**0.64**	0.30	0.54	**0.57**
		58 147	54 96	52 81	64 211	56 104	61 108
	3000	0.34	0.54	**0.64**	0.33	0.54	**0.59**
		71 207	66 122	65 101	72 217	67 125	70 118

3.3.2 Binary classification performance

We report the average maximal AGm (*upper*) and the average maximal Gm (*bottom*) in Table 3.4. The *elastic net*^{AUCPR} most often outperforms the other models at a

Table 3.3: Average actual ranking score. The highest value in each simulation setting is printed in bold. The value with an asterisk is smaller than the result of *elastic net*^{AUCPR} at a significance level of 0.05.

			Ridge			Lasso			Elnet		
τ	n	LR	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR
0.01	800	-	0.12*	0.12*	0.13*	0.19	0.17	0.17	**0.20**	0.19	0.19
	1000	-	0.14*	0.14*	0.14*	0.19	0.18	0.18	**0.21**	0.19	0.20
	3000	0.17	0.31*	0.35*	0.32*	0.41*	0.39*	0.41*	0.42	0.43	**0.44**
0.05	800	-	0.52*	0.52*	0.52*	0.58*	0.57*	0.57*	0.60*	0.61	**0.63**
	1000	-	0.56*	0.56*	0.56*	0.61*	0.61*	0.61*	0.63*	0.64*	**0.66**
	3000	0.48	0.66*	0.71*	0.71*	0.74*	0.75*	0.75*	0.74*	0.75*	**0.78**
0.1	100	-	0.35*	0.35*	0.35*	0.36	0.34*	0.33*	**0.39**	0.37	0.38
	800	-	0.68*	0.68*	0.68*	0.72*	0.72*	0.72*	0.73*	0.75*	**0.77**
	1000	-	0.71*	0.71*	0.71*	0.75*	0.75*	0.75*	0.75*	0.76*	**0.79**
	3000	0.60	0.76*	0.80*	0.80*	0.82*	0.82*	0.82*	0.82*	0.83*	**0.84**
0.2	100	-	0.53*	0.53*	0.53*	0.52*	0.51*	0.50*	0.56	**0.57**	0.55
	800	-	0.80*	0.80*	0.80*	0.82*	0.83*	0.83*	0.83*	0.84*	**0.86**
	1000	-	0.81*	0.81*	0.81*	0.83*	0.84*	0.84*	0.83*	0.84*	**0.86**
	3000	0.71	0.83*	0.87*	0.87*	0.87*	**0.88***	**0.88***	0.87*	**0.88***	**0.88**

significance level of 0.05, in terms of both the maximal AGm and the maxiaml Gm. It also gives the highest TPR and TNR in most of the simulation settings where it is best, due to the fact that the *elastic net*^{AUCPR} performs at least similarly to, if not better than, other models in terms of the (TPR, 1-TNR) pair at almost every threshold. On average, there is not that much difference between the performance of the lasso and elastic net regression, and both of them perform better than the ridge with a rather small advantage. All regularization methods obtain higher maximal AGm and Gm as the data become less imbalanced, and for a larger sample size.

Here, the maximal AGm is often larger than the maximal Gm, and they actually identify the same best and worst model in many simulation settings. Either of them may be used to rank the models. However, they choose different thresholds and different combinations of TPR and TNR. The maximal AGm is often achieved at a slightly higher threshold than the maximal Gm. Therefore, the maximal AGm chooses a lower TPR and a higher TNR than the maximal Gm, as shown in Table 3.5, which presents the average TPR and TNR resulting in the maximal AGm and Gm, respectively. The AGm is more suitable for deciding the threshold for imbalanced data within fraud detection, since the investigations on many false positives can be quite costly.

On average, the optimality measure used in cross-validation of the penalty parameters, rarely affect the maximal AGm of the ridge and lasso regression, except that for $\tau = 0.01$, where the $ridge^{deviance}$ performs much better than $ridge^{AUC}$ and $ridge^{AUCPR}$. This also applies to the TPR and TNR. Usually, all the methods favor the negative class over the positive class by giving higher TNR than TPR. The difference between TNR and

Table 3.4: Average maximal AGm (*upper*) and average maximal Gm (*bottom*). The highest value in each simulation setting is printed in bold. The value with an asterisk is smaller than the result of *elastic net*^{AUCPR} at a significance level of 0.05.

			Ridge			Lasso			Elnet		
τ	n	LR	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR
0.01	800	-	0.74	0.59*	0.45*	0.70	0.75	0.77	**0.79**	0.76	0.78
	1000	-	0.78*	0.72*	0.60*	0.71*	0.77	0.79	**0.81**	0.75	0.80
	3000	0.72	0.86*	0.64*	0.73*	0.88*	0.87*	0.88*	0.88	**0.89**	**0.89**
0.05	800	-	0.86*	0.85*	0.84*	0.88*	0.88*	0.87*	0.89*	0.89	**0.90**
	1000	-	0.88*	0.84*	0.87*	0.89*	0.89*	0.89*	0.90*	0.90*	**0.91**
	3000	0.85	0.91*	0.91*	0.92*	0.93*	0.93*	0.93*	0.93*	**0.94***	**0.94**
0.1	100	-	0.75*	0.73*	0.73*	0.64*	0.74*	0.74*	**0.77**	0.75	0.76
	800	-	0.89*	0.88*	0.89*	0.90*	0.90*	0.91*	0.91*	0.91*	**0.92**
	1000	-	0.90*	0.90*	0.90*	0.91*	0.91*	0.91*	0.92*	0.92*	**0.93**
	3000	0.86	0.92*	0.93*	0.93*	0.94*	0.94*	0.94*	0.94*	0.94*	**0.95**
0.2	100	-	0.77*	0.77*	0.77*	0.77*	0.77*	0.76*	**0.79**	**0.79**	0.78
	800	-	0.90*	0.90*	0.90*	0.92*	0.92*	0.92*	0.92*	0.92*	**0.93**
	1000	-	0.91*	0.91*	0.91*	0.92*	0.93*	0.93*	0.92*	0.93*	**0.94**
	3000	0.87	0.92*	0.94*	0.94*	0.94*	0.94*	0.94*	0.94*	0.94*	**0.95**
0.01	800	-	0.68	0.53*	0.35*	0.63*	0.64*	0.69*	**0.73**	0.67	0.72
	1000	-	0.72	0.70*	0.66*	0.64*	0.68*	0.72	**0.74**	0.69	**0.74**
	3000	0.45	0.83*	0.61*	0.67*	0.84*	0.82*	0.84*	0.84	0.85	**0.86**
0.05	800	-	0.83*	0.80*	0.80*	0.84*	0.84*	0.83*	0.85*	**0.86**	**0.86**
	1000	-	0.84*	0.82*	0.81*	0.86*	0.86*	0.85*	0.86*	0.87*	**0.88**
	3000	0.77	0.89*	0.88*	0.89*	0.91*	0.91*	0.91*	0.91*	**0.92***	**0.92**
0.1	100	-	0.70	0.69	0.68*	0.57*	0.66*	0.66*	**0.71**	0.68	0.70
	800	-	0.86*	0.86*	0.86*	0.88*	0.88*	0.88*	0.88*	0.89*	**0.90**
	1000	-	0.88*	0.86*	0.87*	0.89*	0.89*	0.89*	0.89*	0.90*	**0.91**
	3000	0.81	0.90*	0.91*	0.92*	0.92*	0.92*	0.92*	0.92*	0.92*	**0.93**
0.2	100	-	0.73	0.72	0.72	0.71*	0.70*	0.69*	**0.74**	**0.74**	0.73
	800	-	0.89*	0.88*	0.88*	0.90*	0.90*	0.90*	0.90*	0.91*	**0.92**
	1000	-	0.90*	0.89*	0.89*	0.90*	0.91*	0.91*	0.91*	0.91*	**0.92**
	3000	0.84	0.91*	0.92*	0.92*	**0.93***	**0.93***	**0.93***	**0.93***	**0.93***	**0.93**

TPR is rather large, for highly imbalanced and very high-dimensional data. In such simulation settings, it is impossible to get a high TPR with the AGm.

Figure A.6 (in Appendix A.2) presents the AGm, Gm, TPR and TNR curves of all regularization methods from one simulation, which is run with $n = 1000$ and $\tau = 0.1$. The AGm and Gm curves follow a similar pattern, which can also be observed in the TPR curves, whereas the TNR curves of all the methods are quite similar. As the threshold t increases from 0 to 1, the TPR curves of the ridge reach zero much faster than the TPR curves of the other methods, meaning that the ridge gives more concentrated and much smaller predicted event probabilities to most of the true positives, than the other methods. Hence, the AGm curves of the ridge are very sensitive to the threshold, and a minor alteration in the threshold value will give large changes in the TPR and TNR. The lasso and elastic net regression (except the *elastic net*^{AUCPR})

Table 3.5: $\tau = 0.01$. Average TPR and average TNR according to the average maximal AGm (*upper*) and the average maximal Gm (*bottom*), respectively. The TPR and TNR resulting in the highest average maximal AGm and Gm, are printed in bold, respectively.

n	Ridge			Lasso			Elnet		
	Deviance tpr,tnr	AUC tpr,tnr	AUCPR tpr,tnr	Deviance tpr,tnr	AUC tpr,tnr	AUCPR tpr,tnr	Deviance tpr,tnr	AUC tpr,tnr	AUCPR tpr,tnr
800	0.49 0.87	0.59 0.69	0.62 0.55	0.51 0.83	0.42 0.91	0.45 0.92	**0.50 0.92**	0.49 0.89	0.47 0.92
1000	0.53 0.88	0.61 0.78	0.80 0.55	0.53 0.83	0.48 0.90	0.49 0.92	**0.54 0.92**	0.55 0.86	0.54 0.91
3000	0.67 0.94	0.74 0.69	0.63 0.81	0.69 0.95	0.68 0.94	0.69 0.95	0.70 0.95	**0.72 0.95**	**0.73 0.95**
800	0.71 0.71	0.73 0.58	0.66 0.52	0.68 0.68	0.56 0.79	0.65 0.74	**0.72 0.76**	0.61 0.80	0.70 0.75
1000	0.73 0.73	0.77 0.65	0.86 0.52	0.70 0.69	0.61 0.79	0.67 0.77	**0.72 0.78**	0.70 0.74	**0.70 0.78**
3000	0.82 0.84	0.83 0.63	0.71 0.76	0.82 0.86	0.80 0.85	0.81 0.87	0.83 0.87	0.83 0.87	**0.84 0.87**

behave similarly in the TPR and TNR curves at all thresholds. And their AGm curves are almost flat on the top. Hence, there is a lot of uncertainty as to which threshold value is optimal.

3.3.3 The Brier score: a measure of sharpness

Table 3.6 presents the average Brier score. As the Brier score may be biased towards the negative cases in majority, we present respectively the Brier score bs_1 of the positive class and the Brier score bs_0 of the negative class in Table 3.7, where 0 does not really mean that the average value is zero, since all the results here are with two decimal digits.

On average, the lasso and $elastic\ net^{deviance}$ perform the best with the lowest Brier score, and thus give the sharpest models. The sharpness of the lasso is hardly affected by the measure used in cross-validation of the penalty parameters. The $elastic\ net^{AUC}$ produces much lager Brier score, for $\tau = 0.01$, or $n = 100$, due to a quite large bs_0, which is unusual in the other methods. Also, the ridge performs much worse than the lasso and elastic net, for $\tau = 0.1$ and 0.2, whereas $ridge^{deviance}$ performs often the best among the ridges, in terms of the Brier score. Overall, the optimality measure deviance is better than the AUC and AUCPR, in the sense that it tends to produce smaller Brier score for the ridge and elastic net, and smaller bs_1 for all the methods.

All the methods are much sharper in the negative class than in the positive class with $bs_0 << bs_1$. Also, they (except the $elastic\ net^{AUC}$) have $bs_0 \approx 0$ in many simulation settings. A small bs_0 (with $bs_0 \approx 0$) stems from the fact that most of the true negatives

Table 3.6: Average Brier score. The lowest value in each simulation setting is printed in bold, except for $\tau = 0.01$, where many values are identical.

			Ridge			Lasso			Elnet		
τ	n	LR	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR
0.01	800	-	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.09	0.01
	1000	-	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.12	0.01
	3000	0.01	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.08	0.01
0.05	800	-	0.04	0.05	0.05	**0.03**	**0.03**	**0.03**	**0.03**	0.17	0.04
	1000	-	0.04	0.05	0.05	**0.03**	**0.03**	**0.03**	**0.03**	0.06	0.04
	3000	0.06	0.03	0.05	0.05	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**	0.03
0.1	100	-	0.09	0.09	0.09	**0.08**	**0.08**	**0.08**	0.09	0.20	0.09
	800	-	0.07	0.08	0.09	**0.04**	**0.04**	**0.04**	**0.04**	0.07	0.05
	1000	-	0.07	0.08	0.09	**0.04**	**0.04**	**0.04**	**0.04**	0.05	0.05
	3000	0.09	0.04	0.08	0.09	**0.03**	**0.03**	**0.03**	**0.03**	**0.03**	0.04
0.2	100	-	0.15	0.16	0.16	**0.13**	**0.13**	0.14	**0.13**	0.14	0.14
	800	-	0.12	0.14	0.16	**0.05**	0.06	0.06	**0.05**	0.06	0.07
	1000	-	0.11	0.14	0.16	**0.05**	**0.05**	**0.05**	**0.05**	0.06	0.07
	3000	0.12	0.05	0.13	0.15	**0.04**	**0.04**	**0.04**	**0.04**	**0.04**	0.05

Table 3.7: Average Brier score bs_1 of the positive class and average Brier score bs_0 of the negative class. The lowest bs_1 in each simulation setting is printed in bold.

		Ridge			Lasso			Elnet		
τ	n	Deviance bs_1, bs_0	AUC bs_1, bs_0	AUCPR bs_1, bs_0	Deviance bs_1, bs_0	AUC bs_1, bs_0	AUCPR bs_1, bs_0	Deviance bs_1, bs_0	AUC bs_1, bs_0	AUCPR bs_1, bs_0
0.01	800	0.96 0	0.95 0	0.96 0	0.92 0	0.93 0	0.92 0	0.93 0	**0.88** 0.09	0.94 0
	1000	0.99 0	0.99 0	0.99 0	0.96 0	0.96 0	0.95 0	0.97 0	**0.88** 0.12	0.98 0
	3000	0.94 0	0.95 0.01	0.97 0	**0.75** 0	0.78 0	**0.75** 0	0.78 0	0.81 0.07	0.84 0
0.05	800	0.88 0	0.93 0	0.95 0	**0.60** 0	**0.60** 0	0.62 0	0.63 0	0.63 0.15	0.74 0
	1000	0.86 0	0.92 0	0.95 0	**0.55** 0	0.56 0	0.57 0	0.57 0	0.61 0.04	0.70 0
	3000	0.53 0	0.92 0	0.95 0	**0.35** 0	0.36 0	0.37 0	**0.35** 0	0.39 0	0.51 0
0.1	100	0.91 0.01	0.91 0.01	0.90 0.01	0.83 0.01	0.83 0.01	0.82 0.01	0.83 0.02	**0.77** 0.17	0.85 0.01
	800	0.76 0.01	0.85 0.01	0.89 0.01	**0.39** 0.01	0.41 0.01	0.42 0.01	0.41 0.01	0.48 0.03	0.55 0.01
	1000	0.74 0.01	0.85 0.01	0.90 0.01	**0.36** 0.01	0.38 0.01	0.38 0.01	0.38 0.01	0.43 0.02	0.52 0.01
	3000	0.34 0.01	0.80 0.01	0.87 0.01	**0.24** 0.01	0.25 0.01	0.25 0.01	**0.24** 0.01	0.26 0.01	0.35 0.01
0.2	100	0.80 0.03	0.79 0.05	0.80 0.05	**0.67** 0.04	0.68 0.04	0.69 0.04	0.68 0.03	0.74 0.03	0.73 0.03
	800	0.59 0.03	0.73 0.04	0.79 0.05	**0.26** 0.02	0.27 0.02	0.28 0.02	**0.26** 0.02	0.30 0.02	0.37 0.02
	1000	0.56 0.03	0.71 0.04	0.78 0.05	**0.24** 0.02	0.25 0.02	0.25 0.02	**0.24** 0.02	0.28 0.02	0.34 0.02
	3000	0.23 0.02	0.65 0.04	0.77 0.05	**0.17** 0.02	0.18 0.02	0.18 0.02	**0.17** 0.02	0.18 0.02	0.25 0.01

are given predicted event probabilities $\hat{p}(\boldsymbol{x})s$ that are close to zero. For $\tau = 0.01$, or $n = 100$, most of the models produce very large bs_1 that is close to 0.9, meaning that they assign $\hat{p}(\boldsymbol{x})s$ that are far from one to most of the true positives. Further, Figure A.7 (in Appendix A.2) shows histogram plots of the $\hat{p}(\boldsymbol{x})s$ and true event probabilities of 3000 cases, given by all the regularization methods for $\tau = 0.01$ and $n = 1000$. All the methods assign $\hat{p}(\boldsymbol{x})s$ that are close to zero to most of the cases.

The Brier score of all the methods increases as the data become less imbalanced. This stems from the dominating effect of the increasing bs_0, due to less negative cases in the

training data (i.e., a larger τ). On the other hand, for a smaller sample size, there are fewer positive and negative cases in the training data, thus, larger bs_1, bs_0 (for $\tau = 0.2$) and Brier score are acquired.

3.3.4 The accuracy of predicted event probabilities

Bias and Rmse

Table 3.8 reports the average bias (*upper*) and the average Rmse (*bottom*). Also, we present respectively the average bias b_1 of the positive class and the average bias b_0 of the negative class in Table 3.9.

Table 3.8: Average bias (*upper*) and average Rmse (*bottom*).

			Ridge			Lasso			Elnet		
τ	n	LR	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR
0.01	800	-	0	0.01	0	0	0	0	0	0.08	0
	1000	-	0	0	0	0	0	0	0	0.11	0
	3000	-0.01	0	0.01	0	0	0	0	0	0.07	0
0.05	800	-	-0.01	0	0	-0.01	-0.01	-0.01	-0.02	0.13	-0.01
	1000	-	-0.01	0	0	-0.01	-0.01	-0.01	-0.02	0.02	-0.01
	3000	0.01	-0.02	0	0	-0.01	0	0	-0.01	0	0
0.1	100	-	-0.02	-0.01	0	-0.01	-0.02	-0.01	-0.01	0.11	-0.01
	800	-	-0.01	0	0	-0.02	-0.01	-0.01	-0.02	0	-0.01
	1000	-	-0.01	0	0	-0.02	-0.01	-0.01	-0.02	0	-0.01
	3000	0.03	-0.02	0	0	0	0	0	0	0	0
0.2	100	-	-0.04	-0.01	0	-0.03	-0.04	-0.02	-0.05	-0.05	-0.01
	800	-	-0.01	0	0	-0.02	-0.01	-0.01	-0.02	-0.01	-0.01
	1000	-	-0.01	0	0	-0.01	-0.01	-0.01	-0.02	-0.01	-0.01
	3000	0.03	-0.01	0	0	0	0	0	0	0	0
0.01	800	-	0.09	0.10	0.09	0.09	0.09	0.09	0.09	0.17	0.09
	1000	-	0.10	0.10	0.10	0.09	0.09	0.09	0.09	0.20	0.09
	3000	0.1	0.09	0.10	0.09	0.08	0.08	0.08	0.08	0.15	0.09
0.05	800	-	0.20	0.21	0.21	0.17	0.17	0.17	0.17	0.30	0.19
	1000	-	0.20	0.21	0.21	0.16	0.16	0.16	0.16	0.20	0.18
	3000	0.23	0.16	0.21	0.21	0.13	0.13	0.13	0.13	0.14	0.15
0.1	100	-	0.29	0.29	0.30	0.28	0.28	0.28	0.29	0.38	0.28
	800	-	0.27	0.28	0.29	0.19	0.20	0.20	0.19	0.23	0.23
	1000	-	0.26	0.28	0.29	0.19	0.19	0.19	0.19	0.21	0.22
	3000	0.29	0.18	0.28	0.29	0.15	0.16	0.16	0.15	0.16	0.18
0.2	100	-	0.38	0.39	0.39	0.36	0.36	0.36	0.35	0.36	0.37
	800	-	0.33	0.37	0.39	0.22	0.22	0.23	0.22	0.23	0.26
	1000	-	0.32	0.37	0.39	0.21	0.22	0.22	0.21	0.22	0.25
	3000	0.34	0.21	0.35	0.38	0.18	0.18	0.19	0.18	0.18	0.21

All methods systematically underestimate the event probabilities of true positives, resulting in the b_1 always being negative in Table 3.9. For $\tau = 0.01$ and $n = 3000$, the $b_0 \approx 0$, for methods with the optimality measure deviance. Other than that, all

Table 3.9: Average bias b_1 of the positive class and average bias b_0 of the negative class. The b_1 with the lowest absolute value in each simulation setting is printed in bold.

τ	n	Ridge			Lasso			Elnet		
		Deviance b_1, b_0	AUC b_1, b_0	AUCPR b_1, b_0	Deviance b_1, b_0	AUC b_1, b_0	AUCPR b_1, b_0	Deviance b_1, b_0	AUC b_1, b_0	AUCPR b_1, b_0
0.01	800	-0.88 0.01	-0.88 0.01	-0.88 0.01	-0.86 0.01	-0.87 0.01	-0.86 0.01	-0.86 0.01	**-0.80** 0.09	-0.87 0.01
	1000	-0.94 0.01	-0.94 0.01	-0.94 0.01	-0.90 0.01	-0.89 0.01	-0.90 0.01	-0.91 0.01	**-0.81** 0.12	-0.92 0.01
	3000	-0.88 0	-0.89 0.02	-0.90 0.01	**-0.75** 0	-0.77 0.01	**-0.75** 0.01	-0.77 0	-0.77 0.08	-0.82 0.01
0.05	800	-0.91 0.04	-0.93 0.05	-0.94 0.05	-0.68 0.02	-0.68 0.02	-0.70 0.03	-0.70 0.02	**-0.67** 0.17	-0.82 0.03
	1000	-0.90 0.04	-0.93 0.05	-0.94 0.05	**-0.63** 0.02	**-0.63** 0.02	-0.66 0.02	-0.65 0.02	-0.68 0.06	-0.79 0.03
	3000	-0.62 0.01	-0.93 0.05	-0.95 0.05	**-0.43** 0.02	-0.45 0.02	-0.47 0.02	-0.44 0.02	-0.49 0.02	-0.64 0.03
0.1	100	-0.96 0.09	-0.96 0.10	-0.96 0.11	-0.91 0.09	-0.90 0.08	-0.90 0.09	-0.90 0.09	**-0.81** 0.22	-0.92 0.08
	800	-0.87 0.08	-0.93 0.10	-0.95 0.10	**-0.50** 0.04	-0.52 0.04	-0.55 0.05	-0.52 0.03	-0.60 0.07	-0.71 0.06
	1000	-0.86 0.08	-0.93 0.10	-0.96 0.10	**-0.47** 0.03	-0.49 0.04	-0.51 0.04	-0.48 0.03	-0.57 0.06	-0.68 0.06
	3000	-0.44 0.03	-0.89 0.10	-0.93 0.10	**-0.32** 0.03	-0.35 0.03	-0.36 0.04	**-0.32** 0.03	-0.37 0.04	-0.52 0.05
0.2	100	-0.96 0.17	-0.96 0.22	-0.96 0.24	-0.84 0.16	**-0.82** 0.14	-0.87 0.19	-0.85 0.13	-0.90 0.15	-0.91 0.17
	800	-0.81 0.19	-0.91 0.22	-0.96 0.24	**-0.37** 0.07	-0.41 0.08	-0.43 0.09	-0.38 0.06	-0.46 0.10	-0.60 0.11
	1000	-0.79 0.18	-0.90 0.22	-0.95 0.24	**-0.34** 0.06	-0.38 0.08	-0.40 0.09	-0.35 0.06	-0.43 0.09	-0.56 0.10
	3000	-0.34 0.06	-0.86 0.21	-0.95 0.23	**-0.25** 0.06	-0.28 0.07	-0.29 0.07	-0.26 0.06	-0.29 0.07	-0.43 0.08

methods systematically overestimate the event probabilities of true negatives, resulting in positive b_0.

The bias in Table 3.8 usually has a rather low absolute value, and it is often negative. A negative bias means a systematic underestimation of the true event probabilities for all the cases. Here, it stems from $|b_1| > |b_0|$. As the positives are in minority, a model with a negative bias systematically underestimates the event probabilities of true positives much more than it overestimates the event probabilities of true negatives.

A small Rmse, for instance, for $\tau = 0.01$, means a small spread of the prediction error. As discussed in the results of the Brier score, all the models are rather bad at predicting event probabilities for the positive class for $\tau = 0.01$. But there are only approximately 30 positives in the test set, hence, the Rmse is dominated by the well predicted event probabilities of true negatives in majority.

On average, the deviance is better than the AUC and AUCPR for the ridge and elastic net regression, as the former optimality measure gives relatively lower $|b_1|$, $|b_0|$ and Rmse. The lasso is almost unaffected by the measure used in cross-validation of the penalty parameters, in terms of the bias and Rmse. As the class imbalance decreases, the $|b_0|$ and Rmse increase, due to less negative cases in the training set. For a larger sample size, the $|b_1|$, $|b_0|$ and Rmse decrease. This stems from the fact that the models get better at predicting event probabilities of the two classes, when there are more positive and negative cases in the training set. The results here are in line with the results of the Brier score in Subsection 3.3.3.

Illustration on the probability

Figure 3.1 shows histogram plots of the true and predicted event probabilities of each class, given by all the regularization methods. The simulation is run with $n = 1000$ and $\tau = 0.1$, which represents a normal situation. The true event probabilities of the negative and positive class mainly locate at the left and right endpoint, respectively.

The ridge regression tends to produce predicted event probabilities $\hat{p}(\boldsymbol{x})s$ that are close to the ratio τ of class imbalance. This problem is most severe in $ridge^{AUC}$ and $ridge^{AUCPR}$ with concentrated $\hat{p}(\boldsymbol{x})s$, but alleviated in $ridge^{deviance}$, which produces $\hat{p}(\boldsymbol{x})s$ with a larger spread. The predicted event probabilities given by $ridge^{deviance}$ are closer to the true event probabilities, for each of the two classes, compared to $ridge^{AUC}$ and $ridge^{AUCPR}$. This explains why $ridge^{deviance}$ has slightly lower $|b_1|$, $|b_0|$ and Rmse than the other ridges, for $\tau > 0.01$, or $n > 100$.

Here, there is not that much visual difference between the distributions of $\hat{p}(\boldsymbol{x})s$ that are given by the lasso, $elastic\ net^{deviance}$ and $elastic\ net^{AUC}$. They produce a bimodal distribution of $\hat{p}(\boldsymbol{x})s$ for the true positives, of which many fall into the right most interval (≥ 0.95). Also, most of the true negatives are provided with $\hat{p}(\boldsymbol{x})s$ that fall into the left most interval (< 0.05). This should at least explain why the lasso and $elastic\ net^{deviance}$ usually have the lowest Brier score and RMSE. Compared to $elastic\ net^{deviance}$ and $elastic\ net^{AUC}$, the $elastic\ net^{AUCPR}$ assigns higher $\hat{p}(\boldsymbol{x})s$ to more of the true negatives and lower $\hat{p}(\boldsymbol{x})s$ to more of the true positives.

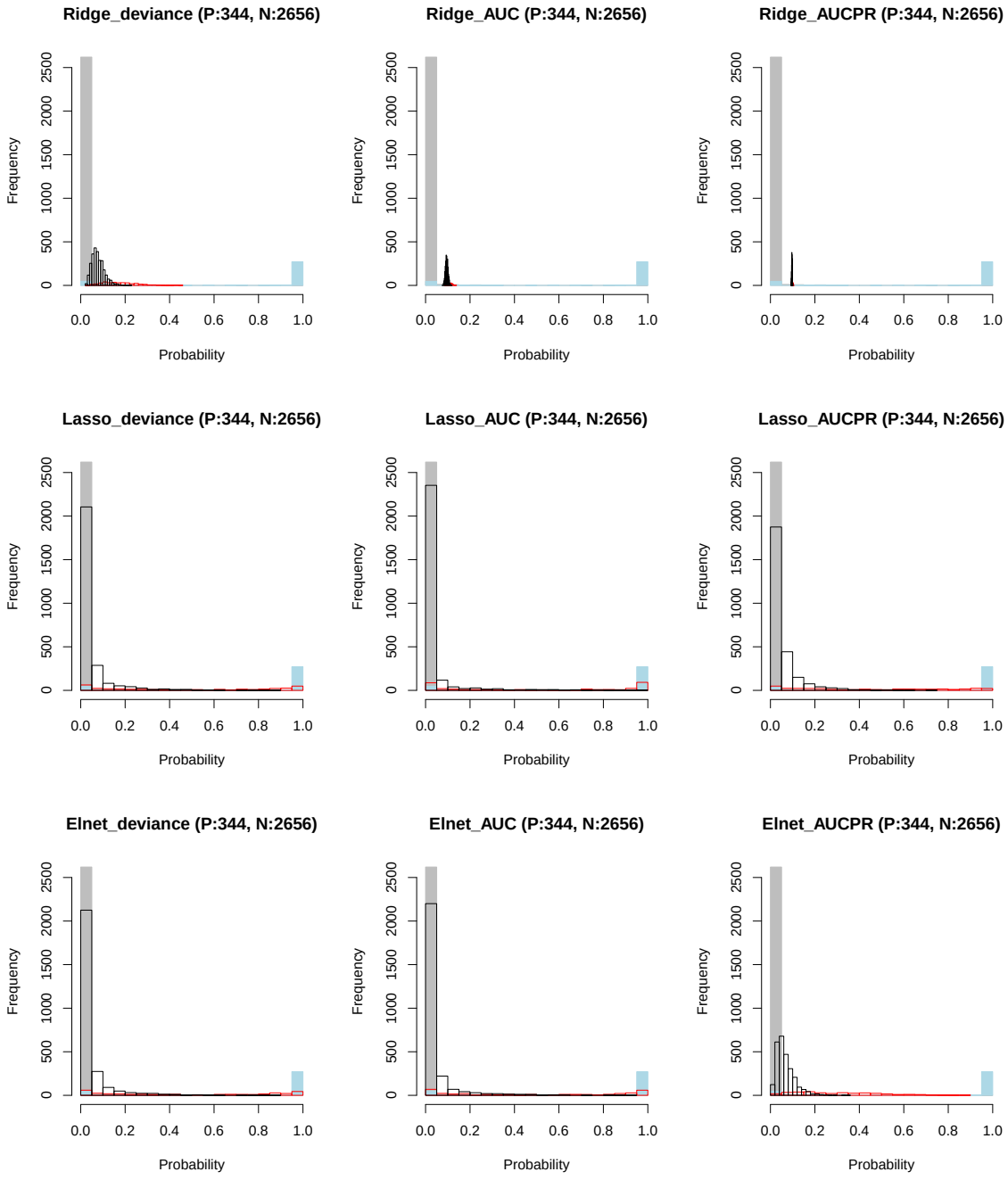


Figure 3.1: Histogram plots of predicted event probabilities of true positives *(red border)*, event probabilities of true positives *(light blue)*, predicted event probabilities of true negatives *(black border)* and event probabilities of true negatives *(grey)*, for $\tau = 0.1$ and $n = 1000$.

Furthermore, reliability diagrams are drawn to check if the regularization methods are well calibrated. In Figure 3.2, the observed event frequency is plotted against the average predicted probability in each bin. A well calibrated model should have the points plotted lie on, or close to, the diagonal. The optimality measure deviance gives a red line that lies closest to the diagonal. All the regularization methods are better calibrated, when using the deviance, rather than the AUC and AUCPR, as the measure in cross-validation of the penalty parameters. In general, all the models severely underestimate the true event probabilities with high values, as the red line lies way above the diagonal in bins to the right, and it is more severe with the optimality measures AUC and AUCPR, but the true event probabilities with high values are few. All the methods slightly overestimate the true event probabilities with low values, which are in majority, and the optimality measure AUC and AUCPR perform slightly worse than the deviance.

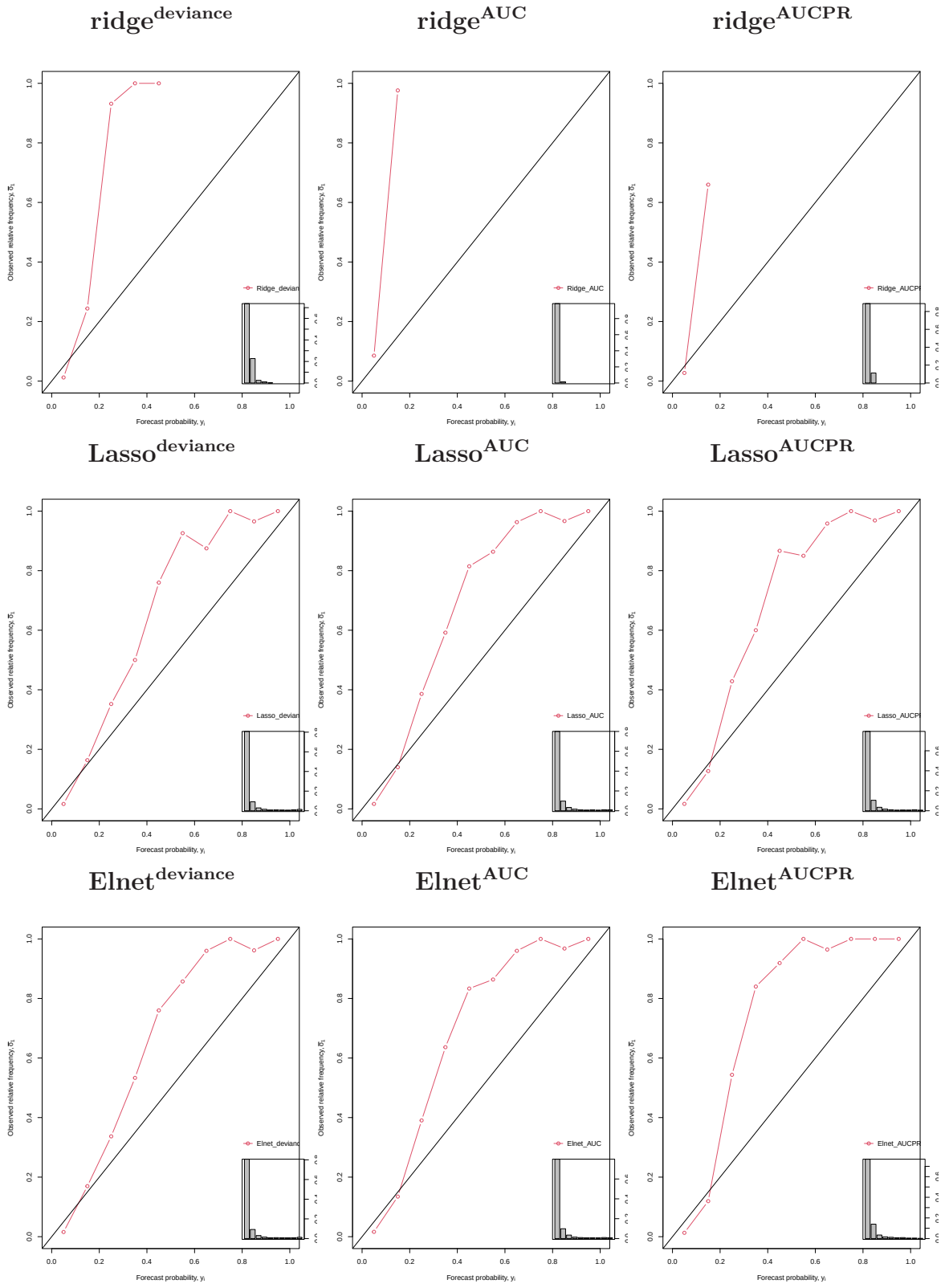


Figure 3.2: Reliability diagrams of the forecast probability plotted against the observed relative frequency, for $\tau = 0.1$ and $n = 1000$. At the bottom right corner of each diagram, a sharpness diagram of the predicted event probabilities for all the cases is plotted.

3.4 Summary

This section aims to summarize the results, and concentrates on drawing conclusions from the simulation study.

The summarized results from this chapter are :

- The $ridge^{deviance}$ performs slightly worse (in terms of the ranking ability) than $ridge^{AUC}$ and $ridge^{AUCPR}$, for class imbalanced data that are not high-dimensional. On the other hand, it produces better predicted event probabilities than the other ridges, in terms of much lower Brier score, Brier score of the positive class (bs_1), Rmse, bias of the positive class (b_1) and bias of the negative class(b_0). In fact, the optimality measure deviance benefits also the lasso and elastic net in achieving lower b_1 and lower b_0 in most of the simulation settings. Nevertheless, all methods tend to systematically overestimate the event probabilities of the true negatives and underestimate the event probabilities of the true positives.

- The prediction performance of the lasso is hardly affected by the optimality measure used in cross-validation of the penalty parameter λ. For instance, the $lasso^{AUC}$, $lasso^{deviance}$ and $lasso^{AUCPR}$ perform equally well in terms of the AUCPR, the AUC, the actual ranking score, the AGm, the Brier score and Rmse. Meanwhile, the $lasso^{AUCPR}$ almost always achieves the highest average proportion of truly influential variables among the selected variables.

- The $elastic\ net^{AUCPR}$ performs the best on average, in terms of the AUCPR, the AUC, the actual ranking score and binary classification performance. It performs also the best among the elastic nets in doing automated variable selection. The performance of the elastic net seems to be boosted by using the AUCPR as the optimality measure in cross-validation of the penalty parameters α and λ. The AUCPR usually chooses an α close to 0, making the elastic net behave more like the ridge, while the deviance and AUC typically choose an α close to 1, making the elastic net behave more like the lasso.

- Non of the methods perform well for highly class imbalanced ($\tau = 0.01$) data, or a very small ratio $\frac{n}{p} = 0.1$. Such simulation settings give rather difficult prediction tasks, where the rather low AUCPR values, rather than the overly optimistic AUC values, strongly express the models' bad ranking ability.

- All penalized logistic regression methods have improved ranking ability (e.g., the AUCPR) and binary classification performance (e.g., the AGm), either for a larger τ (i.e., less imbalance), or a larger ratio $\frac{n}{p}$.

- The improvement in the ranking ability of the models is drastic when the class imbalanced learning problems change from very difficult (e.g., $\tau = 0.01$) to somewhat easier (e.g., $\tau = 0.05$). The improvement in the ranking ability of the models, however, is not significant for the simulation settings that vary from $\tau = 0.1$ to $\tau = 0.2$, while $n \geq p$. In this case, there is a limit to how well the penalized logistic regression methods can predict, despite the fact that the class imbalanced learning problem can become even easier as the τ further increases.
- On average the elastic net performs better than the lasso, which performs on average better than the ridge. This is no surprise as the simulated data are often sparse and with correlated covariates.

Also, the following conclusions are drawn from this chapter:

- Whether the optimality measure used in cross-validation of the penalty parameters affect the prediction performance, depends on the type of learning methods, evaluation metric used for model selection, ratio of class imbalance, sample size, etc.
- Further, when dealing with class imbalanced data, one optimality measure (e.g., the AUC) does not necessarily result in a learning method (e.g., the elastic net) with the optimized performance, in terms of the optimality measure (e.g., the AUC). One needs to be careful with the choice of the optimality measure, for regularized logistic regression approaches in class imbalanced learning scenarios.
- The models that achieve good ranking ability do not necessarily produce sharp predicted event probabilities.
- Which evaluation metric is more appropriate for model selection, depends on the objective of the learning problems. For class imbalanced learning problems within fraud detection, both the AUC and the Brier score are insufficient, since they are biased towards the negative class in majority.
- The AUCPR is strongly recommended for prediction tasks with class imbalanced and high-dimensional data, either for evaluating the average precision of a classifier, or comparing the prediction performance of multiple classifiers. This is important for real fraud detection tasks, as the classifiers that produce decent AUC values may actually perform rather badly in terms of the AUCPR.
- Additionally, the AUCPR, rather than the AUC, can distinctly expose the difference in performance between classifiers when dealing with class imbalanced data. If one is interested in improving the performance of an already good model,

in an attempt of a very good model, the performance measure AUCPR is more appealing than the AUC, since the former may, hopefully, increase much more than the latter does at the same cost of computational complexity.

- As the baseline in the PR space equals the proportion of positives in the data (Saito and Rehmsmeier, 2015), it is a little tricky to use the AUCPR alone as the performance measure for real fraud detection tasks, where the ratio of class imbalance may vary with time, and is unknown for new coming data. Therefore, it is suggested to give the results of both the AUCPR and the AUC for real fraud detection problems with unknown level of class imbalance. This could lead to more reliable conclusions about the models' prediction performance.

3.5 Re-sampling methods

One alternative approach to handle class imbalance is to modify the class distribution of the training data. This can be done by applying re-sampling techniques. In such cases, a more balanced class distribution of the training data is obtained, and the classifier is then fitted to the new sampled and artificial training data. As might be expected, the fitted classifier is then, hopefully, more sensitive to the minority class and its prediction performance on the minority class may be improved. Several studies have demonstrated that re-sampling techniques are effective in improving the performance of classifiers in class imbalanced learning scenarios (Estabrooks et al., 2004; Marques, Garcia, and Sanchez, 2013; Weiss and Provost, 2001).

3.5.1 Random under-sampling and oversampling

There are two major sampling methods including under-sampling and oversampling. The two basic methods can achieve a desired class distribution in the simplest way. For instance, under-sampling randomly eliminates observations from the majority class while keeping all the data of the minority class, whereas oversampling often randomly duplicates observations of the minority class. Obviously, under-sampling reduces the size of the training data, thus, leading to smaller computational complexity. This is, however, the opposite for oversampling.

Regardless of the fact that the two sampling methods alleviate the level of class imbalance in the training data, they have potential problems. Under-sampling may result in losing useful information of the majority class, which could be important for building a good classifier. When there is severe class imbalance with very rare minority cases, under-sampling deletes too many majority cases in order to achieve a desired class distribution, and the resulting training data points may end up being too few to be able to build a good prediction model. As for oversampling, it may still be ineffective

in detecting the positive cases, due to the fact that no new data information of the positive class is actually implemented, if only exact copies of randomly selected positive cases are added in the training data. In the worst case, overfitting may happen when too many identical data points make the decision regions too specific for the minority class, hence, the fitted model probably does not generalize well.

3.5.2 The SMOTE method

Other more sophisticated sampling methods include the synthetic minority oversampling technique (Chawla et al., 2002). The SMOTE may be viewed as an advanced version of oversampling the minority class, as it involves creating synthetic minority class cases rather than random oversampling with replacement (Chawla et al., 2002). The new artificial data points are selected along the line segments between a randomly chosen minority sample and its k nearest neighbors. The SMOTE provides new information of the minority class, such that the decision regions become larger and less specific for the minority class (Chawla et al., 2004). In such cases, overfitting is avoided, and the classifier may generalize better. It is also common to combine oversampling and under-sampling, and Chawla et al. (2002) showed that a combination of SMOTE and under-sampling can achieve better prediction performance, in terms of the AUC, than under-sampling alone.

3.5.3 Earlier work

The effectiveness of re-sampling methods depends on many factors, for instance, the types of learning methods, evaluation metrics (Van Hulse et al., 2007), the number of positive cases in the data (Batista et al., 2004), etc. In Van Hulse et al. (2007), the logistic regression method was applied to several class imbalanced and low-dimensional data sets, and none of the sampling techniques significantly improved the performance of the LR in terms of the AUC. According to Halsteinslid (2019), with the penalized logistic regression methods for class imbalanced data with a large amount of covariates, the re-sampling methods (i.e., under- and oversampling, SMOTE) do not improve much in terms of the AUC. Therefore, in this thesis, we do not experiment with the re-sampling methods for the penalized logistic regression, as it has been done earlier.

3.5.4 Future possibility

Meanwhile, some other directions with the re-sampling methods might be relevant for handling class imbalanced and high-dimensional data. As a perfectly balanced class distribution does not always give optimal results (Weiss and Provost, 2001), one may, therefore, investigate further the relation between the class distribution in the training data and the performance of the penalized logistic regression methods. Hopefully,

there might exist an optimal amount of re-sampling that may give improved prediction performance.

Moreover, since the AUC, rather than the AUCPR, is very insensitive towards the changes in the ratio of class imbalance in the data, it would be interesting to investigate whether the re-sampling methods significantly boost the penalized logistic regression methods in terms of the AUCPR, while the penalized logistic regression methods are optimized with respect to different measures used in cross-validation of the tuning parameters.

Additionally, data cleaning methods that use Tomek links (Tomek, 1976) may be relevant. A Tomek link consists of a pair of points from different classes and are each other's nearest neighbor. Among this pair of points, either one of them is noisy sample, or both of them are borderline examples Batista et al. (2004). One can either remove only the negative case, or both of the points in Tomek links. Therefore, Tomek links methods not only alter the class distribution, but also identify and remove possibly noisy and borderline samples, which may alleviate the overlapping among the classes (Batista et al., 2004). Batista et al. (2004) showed that the SMOTE and Tomek links are recommended for data sets with a small number of positive cases, since the two together present very good AUC results. Hence, one may also investigate if Tomek links, or Tomek links with the SMOTE, would affect the prediction performance of the penalized logistic regression methods in terms of the AUCPR.

Chapter 4

Illustration on credit card default data

In this chapter, we will apply the methods and analysis techniques from the simulation study to a real credit card default data set, which is publicly available in Kaggle.

4.1 Data description and data preprocessing

The credit card default data are not directly fraud data, but they share many of the characteristics of fraud data, such as high dimensionality, class imbalance, and a mixture of discrete and continuous covariates. In addition, this data set contains information of the covariates, which makes data preprocessing easier. It worth noting that real fraud detection data are only available in anonymised form, or in the form of synthetic data sets. The real credit card default data seem to be the best choice for now.

The credit card default data set consists of a dichotomous response and 121 covariates. The ID variable SK_ID_PREV does not affect the outcome, and it is not considered in the modeling process, so only 120 covariates are examined. A case is a default if the outcome $Target$ takes the value 1, and a non-default if the outcome is 0. There are 24825 defaults out of 307511 cases, and the defaults or the positives account for 8.07% of all observations. This value is slightly lower than $\tau = 0.1$ from the simulation study, representing a rather high level of class imbalance. However, the data are not complete, and around 96.31% of observations have missing values. Figure 4.1 shows the proportion of missing data in each of the 120 covariates. Among the missing data, the average proportion of defaults is 0.089, which is only slightly higher than the ratio 0.081 of imbalance in the whole data set.

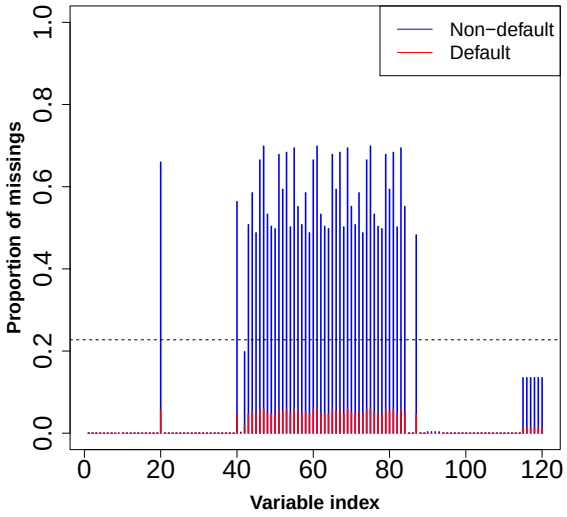


Figure 4.1: The proportion of missing values in each of the 120 covariates. The black horizontal dashed line is the average proportion (22.73%) of missing values. The blue and red histograms represent, respectively, the proportion of non-defaults and defaults among the missing data.

If we discard all the cases with missing values, a lot of information will be lost, and the results may not be representative of the true underlying data pattern. As the penalized logistic regression methods do not deal with missing values automatically, imputation methods are used to preserve and to make best use of all the observations. Imputation methods replace the missing data with appropriate values that are based on the available information.

For instance, for numerical data, we simply replace a missing value with the median of that variable, which is computed with the non-missing data of the other observations. To distinguish original data from imputed data, a new indicator variable is generated for each numerical variable with missing data. It equals one for a particular observation if the value of that numerical variable is imputed, and zero, otherwise. There are 61 numerical covariates with missing data, so the additional indicator variables from median imputation increase the number of covariates from 120 to 181.

As for categorical variables, a missing value is simply replaced with the missing category. The missing level may serve as a new dummy variable of that categorical variable in the predictive model, which may affect the prediction performance if the missing mechanism is related to the outcome.

An illustration of the imputation practice is provided in Table 4.1. The substitution is straightforward and easy to implement on the credit card default data set, which has a large size. Unfortunately, this simple imputation process may introduce some bias. Other more sophisticated imputation methods may be used, but it is not the objective of this study to explore different imputation methods.

Table 4.1: Imputation methods. Here, X_1 is a numerical variable, and X_2 a categorical variable.

X_1	X_2
3090	$level_1$
449	NA
NA	$level_2$
1550	$level_3$

$\Longrightarrow$

X_1	$X_1, missing$	X_2
3090	0	$level_1$
449	0	$missing$
1550	1	$level_2$
1550	0	$level_3$

Among the 181 explanatory variables, 72 are numerical and 109 are categorical. For a categorical variable with k levels and $k > 2$, we convert it into $k - 1$ dummy variables. The reason is that one level will be captured by the intercept, and it is specified when all the other $k - 1$ dummy variables are set to zero. Therefore, 115 additional variables are introduced, which results in a total number of 296 covariates. Lastly, 41 variables are dropped due to that, either they are perfectly linearly dependent with other variables, or they have the same value for almost all the observations. The final number p of covariates is then 255.

4.2 Setup

Compared to the size 307511 of the credit card default data set, the number 255 of covariates is relatively small. Unfortunately, the credit card default data set does not meet the requirement that fraud data are possibly high-dimensional (i.e., $n \leq p$ in this thesis). Therefore, we will sample stratified subsets from the original data set. The resulting subsets will have different ratios τ of class imbalance and varying ratios between the sample size n and the number p of covariates, in order to explore different model settings.

The ratio τ of class imbalance in both the training set and the test set takes the values of 0.05, 0.1 and 0.2. The sample size n is set to be 100, 255, or 1000, leading to the ratio $\frac{n}{p}$ values of 0.39, 1 and 3.92, respectively. The sample size of $n = 1000$ is only used as a reference, since it is low-dimensional with $n > p$. The combination of $\tau = 0.05$ and $n = 100$ is not included, since there will be approximately five positive cases in the sampled training set, which is very difficult for performing cross-validation. Therefore, we have in total 8 model settings. The expected number of defaults or positive cases in

the training set is no more than 20 in most of the model settings. To avoid convergence issues, stratified five-fold cross-validation in Subsection 2.5.2 is implemented for selecting the optimal values of the penalty parameters.

The test set is sampled independently of the training set, and its size m is set to be 5000, which is large enough, such that the prediction performance is representative. The sampling of a training set and a test set is repeated 100 times for each model setting. As the main focus of the study is evaluation metrics, and the effect of different optimality measures (used in cross-validation of the penalty parameters) on the prediction performance of the penalized logistic regression methods, we do not sample a third independent data set to assess the performance of the final chosen model.

Further, we use purely training data to identify the median of a variable for imputation, so the missing numerical data in both the training set and the test set are replaced with the median of that variable obtained from the training set. Moreover, numerical covariates in the test set are standardized with the mean and standard deviation of that covariate computed from the training set. By doing this, we avoid data leakage from the training data to the test data.

4.3 Results

In this section, we present the results of the penalized logistic regression methods, which are applied to stratified subsets of the credit card default data. The prediction performance of all the methods is evaluated with the following measures: the AUCPR, the AUC, the maximal AGm with corresponding TPR and TNR, the maximal Gm with corresponding TPR and TNR, and the Brier score.

For $n \leq p$ with $\tau = 0.05$ and 0.1, the lasso and elastic net regression only manage to converge for around 50% and 75% of the 100 sampled data sets, respectively. The ridge is quite powerful, in the sense that it almost has no problem in converging, regardless of the ratio τ of class imbalance and the sample size n. Only the results from the converged models are considered.

4.3.1 Ranking ability

Table 4.2 presents the average AUCPR and AUC. On average the lasso and elastic net perform better than the ridge, and there is not that much difference between the overall ranking ability of the lasso and elastic net. Among the lassos, $lasso^{AUCPR}$ most often produces the highest AUCPR and AUC. This also applies to $elastic\ net^{AUCPR}$ among the elastic nets, where the optimality measure AUCPR boost the AUC of the

Table 4.2: Average AUCPR (*upper*) and average AUC (*bottom*). The highest value in each model setting is printed in bold.

		Ridge			Lasso			Elnet		
τ	n	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR
0.05	255	0.072	0.072	0.071	0.070	**0.082**	0.081	0.073	0.077	0.078
	1000	0.085	0.085	0.083	0.104	0.103	**0.107**	0.099	0.099	0.101
0.1	100	0.133	0.132	0.132	0.122	0.126	0.130	**0.136**	0.130	0.133
	255	0.149	0.149	0.147	0.163	**0.168**	0.167	0.161	0.166	0.167
	1000	0.179	0.178	0.175	0.214	0.214	0.216	0.211	0.214	**0.217**
0.2	100	0.266	0.267	0.263	**0.282**	0.278	0.278	0.278	0.280	0.281
	255	0.297	0.296	0.295	0.333	0.325	0.333	0.329	0.325	**0.334**
	1000	0.347	0.346	0.336	0.390	0.390	**0.392**	0.389	0.389	0.391
0.05	255	0.604	0.603	0.599	0.574	0.631	**0.632**	0.602	0.615	0.627
	1000	0.643	0.639	0.632	0.679	0.675	**0.685**	0.674	0.674	0.682
0.1	100	0.586	0.584	0.582	0.549	0.566	0.580	0.592	0.581	**0.594**
	255	0.624	0.622	0.616	0.641	0.647	0.653	0.646	0.649	**0.655**
	1000	0.669	0.666	0.651	0.704	0.700	0.703	0.702	0.701	**0.705**
0.2	100	0.603	0.603	0.596	0.619	0.615	0.619	0.621	0.621	**0.625**
	255	0.639	0.636	0.628	0.670	0.659	0.670	0.668	0.661	**0.671**
	1000	0.690	0.688	0.665	**0.718**	0.716	0.715	0.717	0.716	0.716

elastic net to a large extent. Still, the advantage with the optimality measure AUCPR is only observed in the third decimal. Among the ridges, the $ridge^{deviance}$ most often performs the best in terms of the AUC in the third decimal. Nevertheless, none of the regularization methods is significantly affected by the optimality measure used in cross-validation of the penalty parameters.

All the methods have better performance as the data become less imbalanced, and for a larger sample size. However, they all have poor performance in terms of both the AUCPR and the AUC, especially for $\tau = 0.05$ and $n = 100$. In general, the AUCPR is not much higher than the ratio τ of class imbalance, and the AUC, for $n \leq p$, only varies from around 0.57 to 0.67. Therefore, none of the models predict much better than a random guessing classifier. The learning from the credit card default data must be much more difficult than learning from the simulated data in Chapter 3. This is no surprise, as the tasks of learning from real imbalanced data are almost never easy. Also, the AUC is rather low, even for much less imbalanced data with $\tau = 0.2$. There might be a lot overlapping among the classes in the credit card default data, or other methods rather than logistic models may fit this imbalanced data set better, for instance, random forest, neural networks, etc.

In this case, the AUC can be used for comparing the performance of multiple classifiers, due to the fact that the AUC, for $n \leq p$, actually reveals the poor ranking ability of all

the models through values that are hardly higher than 0.65. The AUCPR is, however, seen to be a more appropriate measure if the positive class is more important than the negative class for the decision maker, which is the case in fraud detection. Besides, for the lasso and elastic net regression, the results of AUCPR is less variable than the AUC. In Figure A.8 (in Appendix A.3), a much smaller standard deviation is observed in the AUCPR *(left)* than in the AUC *(right)*.

4.3.2 Binary classification performance

It is observed in Table 4.3 that the maximal AGm and Gm often select the same best model. In this case, if the goal is to compare multiple classifiers in order to select the best one, which of the AGm and GM to choose, does not matter.

Table 4.3: Average maximal AGm (*upper*) and average maximal Gm (*bottom*). The highest value in each model setting is printed in bold.

		Ridge			Lasso			Elnet		
τ	n	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR
0.05	255	0.41	0.54	0.36	0.57	0.63	**0.66**	0.62	0.59	0.64
	1000	0.61	0.60	0.53	0.62	0.68	**0.70**	0.60	0.65	0.67
0.1	100	0.58	0.58	0.52	0.62	0.63	0.63	**0.64**	0.59	**0.64**
	255	0.54	0.61	0.49	0.64	0.68	**0.69**	0.67	0.63	0.68
	1000	0.66	0.66	0.61	0.68	0.69	**0.70**	0.69	0.68	**0.70**
0.2	100	0.59	0.61	0.54	0.65	0.65	**0.66**	**0.66**	0.64	0.65
	255	0.62	0.64	0.61	0.68	0.68	**0.69**	0.68	0.66	**0.69**
	1000	0.67	0.67	0.65	**0.70**	**0.70**	**0.70**	**0.70**	**0.70**	**0.70**
0.05	255	0.41	0.51	0.42	0.49	0.54	**0.56**	0.53	0.51	0.55
	1000	0.57	0.56	0.55	0.55	0.60	**0.62**	0.53	0.58	0.59
0.1	100	0.55	0.55	0.54	0.42	0.50	0.52	**0.56**	0.50	0.55
	255	0.49	0.56	0.50	0.56	0.61	0.61	0.60	0.58	**0.62**
	1000	0.60	0.60	0.57	0.63	0.63	**0.64**	0.63	0.63	**0.64**
0.2	100	0.56	0.56	0.54	0.58	0.58	**0.60**	**0.60**	0.58	**0.60**
	255	0.57	0.58	0.53	**0.63**	**0.63**	**0.63**	**0.63**	0.60	**0.63**
	1000	0.62	0.62	0.59	**0.65**	**0.65**	**0.65**	**0.65**	**0.65**	**0.65**

On average the ridge produces lower AGm than the lasso and elastic net, and $ridge^{AUCPR}$ performs the worst among the ridges. The $lasso^{AUCPR}$ out of all the methods most often gives the highest AGm, whereas $lasso^{deviance}$ gives much lower AGm than the other lassos, for $\tau = 0.05$. The $elastic\ net^{AUCPR}$ most often gives the highest AGm among the elastic nets, but the difference between the performance of $elastic\ net^{AUCPR}$ and $elastic\ net^{deviance}$ is negligible, for $\tau = 0.1$ and 0.2. For the lasso and elastic net, the optimality measure AUCPR may be preferred, since it on average gives better results than the deviance and AUC.

The maximal AGm is obtained at a higher threshold than the maximal Gm. The latter usually chooses the optimal threshold that is close to τ. Therefore, the AGm achieves a higher TNR and a lower TPR than the Gm, as shown in Table 4.4. Even though the TNR obtained with the maximal AGm is very high, indicating a relatively small FPR (loss), the TPR (benefit) is less than 0.5. The best model chosen by the maximal Gm gives a balanced combination of TPR and TNR, whose values vary only from 0.5 to 0.6. Regardless of whether the AGm or the Gm is chosen to decide the optimal threshold for classification, the resulting TPR will not be very high. The TPR only achieves a high value at a much smaller threshold than τ, as shown in Figure A.9 (in Appendix A.3), where we observe also that rather small predicted event probabilities are assigned to most of the cases. As there is a very large proportion of negative cases in the data, a rather small FPR is desired in fraud detection. Therefore, the AGm is still recommended for selecting the optimal threshold for such high-dimensional and class imbalanced data.

Table 4.4: $\tau = 0.1$. Average TPR and average TNR according to the average maximal AGm (*upper*) and the average maximal Gm (*bottom*), respectively. The TPR and TNR resulting in the highest average maximal AGm and Gm, are printed in bold, respectively.

n	Ridge			Lasso			Elnet		
	Deviance tpr,tnr	AUC tpr,tnr	AUCPR tpr,tnr	Deviance tpr,tnr	AUC tpr,tnr	AUCPR tpr,tnr	Deviance tpr,tnr	AUC tpr,tnr	AUCPR tpr,tnr
100	0.37 0.71	0.37 0.72	0.47 0.60	0.20 0.90	0.24 0.86	0.22 0.87	**0.30 0.82**	0.30 0.78	**0.28 0.83**
255	0.43 0.67	0.34 0.78	0.32 0.72	0.36 0.81	0.33 0.85	**0.35 0.85**	0.35 0.83	0.36 0.80	0.35 0.84
1000	0.33 0.83	0.34 0.81	0.25 0.85	0.39 0.83	0.36 0.85	**0.38 0.85**	0.38 0.84	0.39 0.82	**0.38 0.85**
100	0.59 0.52	0.59 0.52	0.61 0.49	0.32 0.79	0.47 0.67	0.49 0.65	**0.55 0.60**	0.51 0.61	0.55 0.60
255	0.67 0.46	0.62 0.54	0.75 0.35	0.62 0.60	0.60 0.63	0.59 0.65	0.61 0.62	0.66 0.56	**0.62 0.62**
1000	0.61 0.59	0.60 0.60	0.64 0.51	0.64 0.63	0.62 0.64	**0.64 0.64**	0.63 0.64	0.64 0.63	**0.64 0.64**

4.3.3 The Brier score

Table 4.5 reports the average Brier score. Table 4.6 presents the average Brier score bs_1 of the positive class and the average Brier score bs_0 of the negative class.

All the methods produce almost identical Brier score in each model setting. The fact that there is not that much difference between the bs_0 of all the methods in each model setting, explains why similar average Brier scores are produced, since the negative cases are in majority, and the total Brier score is dominated by the bs_0.

Table 4.5: Average Brier score.

		Ridge			Lasso			Elnet		
τ	n	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR
0.05	255	0.05	0.05	0.05	0.05	0.06	0.05	0.05	0.05	0.05
	1000	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
0.1	100	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09
	255	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09
	1000	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09
0.2	100	0.16	0.16	0.16	0.16	0.17	0.16	0.16	0.17	0.16
	255	0.16	0.16	0.16	0.15	0.16	0.15	0.15	0.16	0.15
	1000	0.15	0.15	0.16	0.14	0.15	0.15	0.14	0.15	0.15

Table 4.6: Average Brier score bs_1 of the positive class and average Brier score bs_0 of the negative class. The lowest bs_1 in each model setting is printed in bold.

		Ridge			Lasso			Elnet		
τ	n	Deviance bs_1, bs_0	AUC bs_1, bs_0	AUCPR bs_1, bs_0	Deviance bs_1, bs_0	AUC bs_1, bs_0	AUCPR bs_1, bs_0	Deviance bs_1, bs_0	AUC bs_1, bs_0	AUCPR bs_1, bs_0
0.05	255	0.89 0	0.89 0	0.90 0	0.89 0	**0.88** 0.01	**0.88** 0	0.89 0	0.90 0	**0.88** 0
	1000	0.89 0	0.88 0	0.90 0	0.87 0	0.88 0	**0.86** 0	0.87 0	0.88 0	0.87 0
0.1	100	0.80 0.01	0.80 0.01	0.81 0.01	0.79 0.01	0.79 0.01	**0.78** 0.01	**0.78** 0.01	0.80 0.01	0.79 0.01
	255	0.79 0.01	0.79 0.01	0.80 0.01	0.77 0.01	0.77 0.02	**0.76** 0.01	0.78 0.01	0.78 0.01	0.77 0.01
	1000	0.77 0.01	0.77 0.01	0.80 0.01	**0.75** 0.01	**0.75** 0.01	**0.75** 0.01	**0.75** 0.01	0.77 0.01	0.76 0.01
0.2	100	0.63 0.04	0.63 0.04	0.64 0.04	**0.61** 0.04	0.62 0.06	**0.61** 0.05	**0.61** 0.04	0.63 0.05	0.62 0.04
	255	0.61 0.04	0.62 0.04	0.64 0.04	**0.59** 0.04	**0.59** 0.05	**0.59** 0.04	**0.59** 0.04	0.60 0.05	0.60 0.04
	1000	0.57 0.04	0.58 0.04	0.63 0.04	**0.55** 0.04	0.56 0.04	0.56 0.04	**0.55** 0.04	0.58 0.04	0.58 0.04

On average, the lasso and $elastic\ net^{deviance}$ give lower bs_1 than the other methods, but the difference is not that much. The $ridge^{AUCPR}$ gives usually slightly higher bs_1 than $ridge^{deviance}$ and $ridge^{AUC}$, whereas the $lasso^{AUCPR}$ most often produces the lowest bs_1 among all the methods. However, for the lasso and elastic net, the difference between the bs_1 resulted from different optimality measures is rather minor.

The average Brier score increases for a larger τ, meaning that a model is less sharp and less confident about its prediction as the data become less imbalanced. It is worth noting that, it does not necessarily mean that the model is worse in prediction performance, for instance, in terms of the ranking ability. And the reason why the Brier score increases for less imbalance, is that a fitted model becomes less sharp in the negative class, as a consequence of much more positives or defaults in the training data. So the fitted model produces a slightly higher bs_0, which dominates the total Brier score due to a very large number N of negative cases in the test set, despite the fact that the model is actually much sharper in producing predicted event probabilities that are closer to 1 for true positives. As observed in Table 4.6, the bs_1 actually decreases much more than the bs_0 increases, for a larger τ.

4.3.4 Tuning the penalty parameter

As discussed earlier, the ranking ability of each of the regularization methods is not significantly affected (not until the third decimal) by the optimality measure used in cross-validation of the penalty parameters. However, for the ridge regression, the AUCPR tends to choose a much larger λ than the deviance and AUC, as shown in Figure A.10 (in Appendix A.3). Too much penalization may be the reason why the $ridge^{AUCPR}$ most often produces lowest AUCPRs and AUCs in third decimal. Additionally, the optimal value of λ decreases for a larger sample size. As shown in Figure A.11 (in Appendix A.3), for the lasso regression, the optimal value of λ chosen by each of the three optimality measures is quite similar, which directly explains why the lasso is not affected by the measure used in tuning the penalty parameter.

Moreover, it is of interest to investigate whether tuning the penalty parameter affects the prediction performance of the regularization methods, for the credit card default data. Figure 4.2 shows respectively the cross-validated AUC curves *(upper panel)* and the cross-validated AUCPR curves *(bottom panel)* as the functions of the penalty parameter λ for the ridge, lasso and elastic net ($\alpha = 0.4$) regression. The curves are plotted from one random experiment run with $\tau = 0.1$ and $n = 255$. All curves are concave and have a maximum point, meaning that tuning the penalty parameter is not meaningless. In the ridge regression, the cross-validated AUCPR curve is quite flat over a wide range near the maximum, whereas the cross-validated AUC curve is pointed around the maximal AUC. This explains why $ridge^{AUCPR}$ produces much smaller AUC, but only slightly lower (hardly noticeable) AUCPR than the other ridges, since the AUCPR usually chooses a larger λ than the AUC and deviance in Figure A.10 (in Appendix A.3). It is also observed that the AUCPR and AUC choose approximately the same optimal value of λ for the lasso and elastic net ($\alpha = 0.4$), which again gives almost the same ranking ability, either evaluated by the AUC, or the AUCPR.

The elastic net tends to select many more covariates than the lasso, as shown in Figure A.12 (in Appendix A.3). Although the lasso selects fewer variables, it still performs similarly (in terms of the ranking ability) as the elastic net that selects many more variables. The reason may be that, compared to the elastic net, the lasso selects variables which consist of adequate truly influential variables. In terms of interpretation, the simpler model is typically better. The lasso should be favored over the elastic net in terms of model interpretation. The $lasso^{deviance}$ and $elastic\ net^{AUCPR}$ select respectively the least amount of variables among the lassos and elastic nets. No particular relation is observed between the number of selected variables and the prediction performance in terms of the AUCPR and AUC.

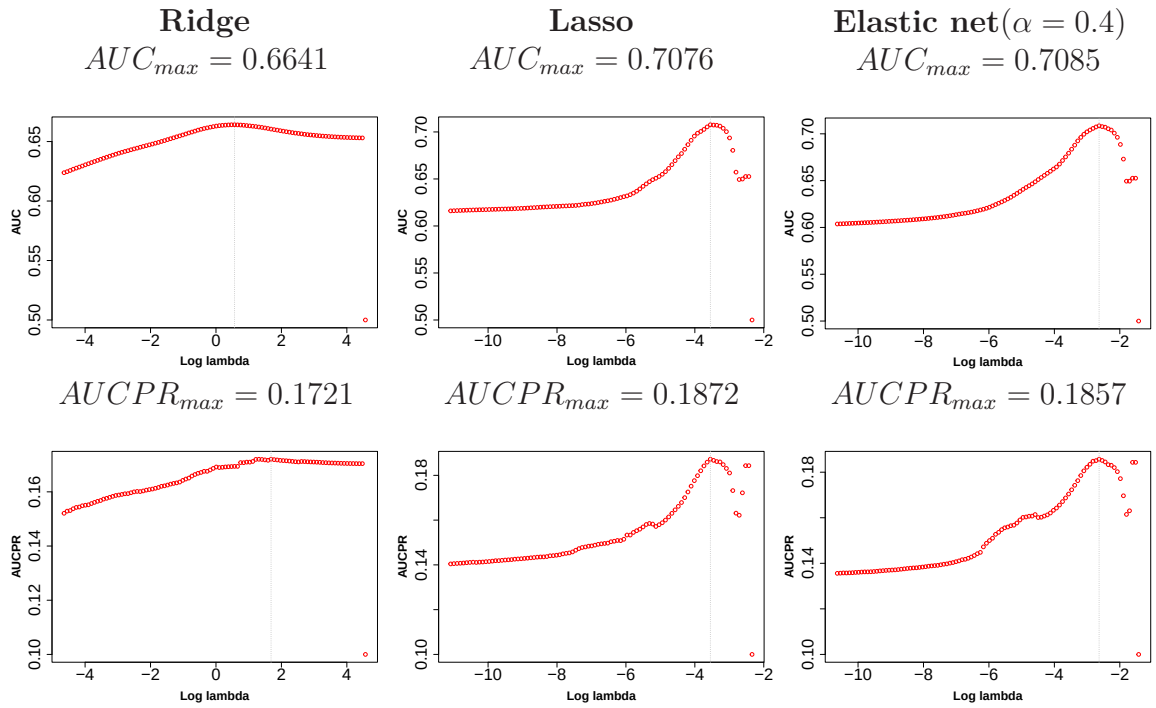


Figure 4.2: Cross-validation plots of how the AUC *(upper panel)* and AUCPR *(bottom panel)* vary with the penalty parameter λ in the ridge *(left)*, the lasso *(middle)* and elastic net *(right)* regression, for $\tau = 0.1$ and $n = 255$. The grey dashed line marks the λ value where the best ranking ability is achieved. The AUC and AUCPR actually select respectively the same optimal value of λ in the lasso and elastic net ($\alpha = 0.4$) regression. Models do not converge for a too big penalty, therefore, producing an abnormal red point that drops to the baseline level.

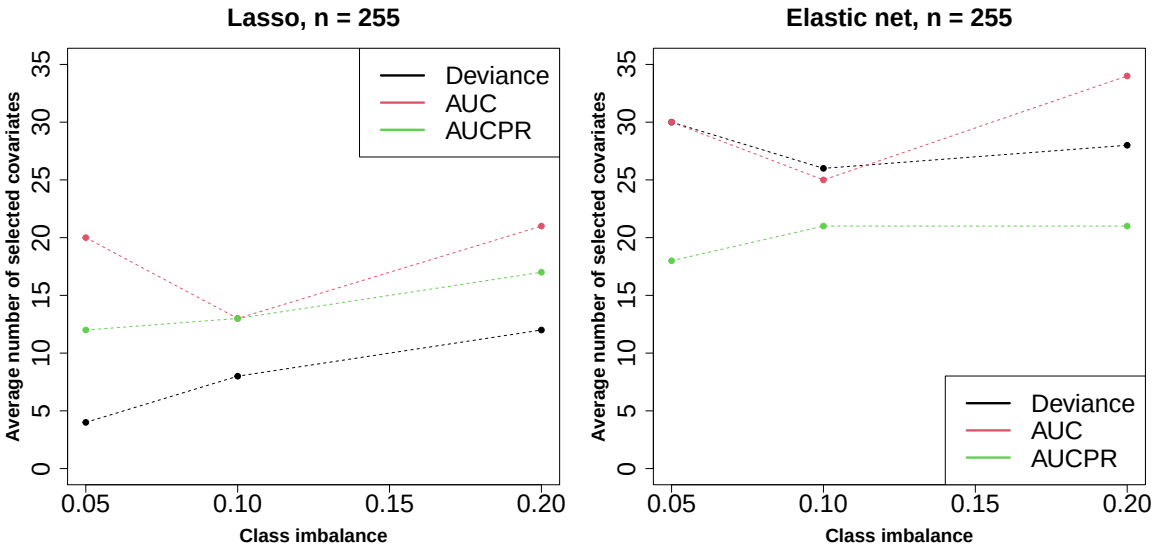


Figure 4.3: Plots of average number of selected covariates in the lasso and elastic net regression, respectively. The sample size $n = 255$, which is also the number p of covariates.

4.3.5 An additional evaluation metric

Even if the models' actual ranking ability measured by the actual ranking score cannot be examined here, it is still possible to calculate the proportion of true positives among the top n_1 cases with the highest predicted event probabilities. The n_1 is the number of the true positives in the test set. Like the actual ranking score, this alternative measure is also based on the assumption that the insurance company is only capable of manually controlling a few cases that are most likely to be fraudulent, based on the predicted event probabilities $\hat{p}(\boldsymbol{x})s$ returned by the prediction model. Further, whether the $\hat{p}(\boldsymbol{x})$ is well-calibrated or not, does not affect the ordering among all the cases according to the $\hat{p}(\boldsymbol{x})$. This additional measure should also equal the proportion of successfully detected true positives among all the true positive cases. In other words, it corresponds to the particular TPR determined by a threshold which gives n_1 predicted positive cases.

Table 4.7 reports the average proportion of true positives among the top n_1 cases with the highest $\hat{p}(\boldsymbol{x})s$. On average, the lasso and elastic net regression perform better than the ridge. The ridge preforms the same, regardless of the optimality measure used for tuning the penalty parameter. This also applies to the lasso and elastic net regression for $n = 1000$. The $lasso^{deviance}$ always gives the highest score, and it outperforms any other method substantially, especially for very high-dimensional and severely class imbalanced data. It is not easy to decide which optimality measure is the best or the worst for the elastic net, as it seems to be sample-dependent, at least for high-dimensional data.

Table 4.7: Average proportion of true positives among the n_1 cases with the highest predicted event probabilities. The highest value in each model setting is printed in bold.

		Ridge			Lasso			Elnet		
τ	n	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR	Deviance	AUC	AUCPR
0.05	255	0.09	0.09	0.09	**0.43**	0.20	0.11	0.30	0.18	0.11
	1000	0.11	0.11	0.11	**0.15**	0.14	**0.15**	0.14	0.14	0.14
0.1	100	0.15	0.15	0.15	**0.49**	0.27	0.18	0.16	0.23	0.15
	255	0.18	0.18	0.18	**0.26**	0.22	0.22	0.23	0.21	0.21
	1000	0.22	0.22	0.22	**0.26**	**0.26**	**0.26**	**0.26**	**0.26**	**0.26**
0.2	100	0.29	0.29	0.29	**0.36**	0.31	0.30	0.33	0.31	0.31
	255	0.33	0.33	0.33	**0.36**	0.35	**0.36**	0.35	0.35	**0.36**
	1000	0.38	0.38	0.36	**0.41**	**0.41**	**0.41**	**0.41**	**0.41**	**0.41**

Compared to the AUC which looks at the overall ranking ability, the proportion of true positives among the n_1 cases with the highest predicted event probabilities, is more related to the practical purpose of detecting the few true positives among a large number of cases. It is also related to ranking, but only of a subset of the cases.

4.4 Summary

In this chapter, we applied the penalized logistic regression methods to a real credit card default data set.

The results from this chapter are summarized below:

- The $ridge^{AUCPR}$ performs the worst among the ridges, in terms of the AUCPR and AUC, and its worst performance is, however, only observed in the third decimal. Other than that, the prediction performance of the ridge is hardly affected by the optimality measure used in cross-validation of the penalty parameter λ.
- On average, the $lasso^{AUCPR}$ and $elastic\ net^{AUCPR}$ perform respectively the best among the lassos and among the elastic nets, especially when they are evaluated in terms of the AUCPR, the AUC, the AGm and the Gm. In particular, the elastic net is boosted by the optimality measure AUCPR, as it very often achieves the highest AUC and AUCPR among all the methods. Nevertheless, the difference between the classifiers' performance in terms of the AUCPR and the AUC is only significant in the third decimal.
- Overall, all the methods show rather poor ranking ability, especially for highly class imbalanced, or very high-dimensional data (i.e., $\tau = 0.05$, or $n = 100$). The imbalanced learning from the credit card default data is much more difficult than the simulation study in Chapter 3. Here, the AUC reveals also the models' poor performance. In this case, either of the AUCPR and the AUC may be used as the performance measure for model selection.
- The lasso and elastic net perform better than the ridge, in terms of both the overall ranking ability and the binary classification performance. In each setting, all the methods produce rather similar Brier score bs_0 of the negative class, and, therefore, obtain almost identical total Brier score. The difference in the Brier score bs_1 of the positive class is negligible, either between the penalized logistic regression methods, or between the optimality measures used in cross-validation of the penalty parameters.
- As the data become less imbalanced, or for a larger sample size n, all the methods have better performance, in terms of larger AUCPR, AUC, AGm and Gm, and smaller Brier score bs_1 of the positive class.
- The $lasso^{deviance}$ significantly outperforms any other method, in achieving the highest proportion of true positives among the top-ranked n_1 cases according the

predicted event probabilities, especially for high imbalance (i.e., $\tau = 0.05$), or high-dimension (i.e., $n = 100$).

The following conclusions are drawn:

- The illustration on the credit card default data produces some different results compared to the simulation study in Chapter 3. Still, we may conclude that, whether the optimality measure used in cross-validation of the penalty parameters is effective in selecting the optimal penalty parameter value that will optimize the prediction performance, or if there should be any preference for the optimality measure, seems to depend on the type of learning methods and data, evaluation metric used for model selection, etc.
- When the imbalanced learning problem gets very difficult, the AUC may reveal the poor ranking ability of a model, and thus, may be used to rank multiple models in model selection. But the AUC may still not suit fraud detection, where detecting the positive class is usually of the main interest.
- The Brier scores of both classes (i.e., bs_1 and bs_0) are suggested to be investigated, in addition to the Brier score, for imbalanced learning problems.
- In this case, if one is even interested in the improvement in terms of the AUCPR and AUC in the third decimal, the AUCPR is then suggested as the optimality measure in cross-validation of the penalty parameters, for the elastic net and lasso regression.

Chapter 5

Conclusion and discussion

In this thesis, we have used the penalized logistic regression models for handling class imbalanced and high-dimensional data, and explored different measures of prediction performance used for model selection in the setting of fraud detection. In particular, a large part of the work has been devoted to investigating the behavior of the penalized logistic regression models for different optimality measures used in cross-validation of the penalty parameters, and to demonstrating the evaluation metrics that are appropriate, or inappropriate for model selection when the data are class imbalanced, and possibly high-dimensional.

This has been investigated on simulated data and on sample subsets of a real credit card default data, with a range of imbalance levels, and with varying ratios between the sample size and the number of covariates.

We have demonstrated that, for class imbalanced and high-dimensional data, the AUCPR has many advantages over the AUC, both in comparing the performance of multiple classifiers and in giving an accurate interpretation of a classifier's prediction performance. We recommend that the AUCPR, rather than the AUC, is of more practical use for fraud detection, as it, when necessarily, can manage to reveal the poor performance of a classifier on the minority class.

Furthermore, we discovered that, for data that are neither severely imbalanced nor very high-dimensional, the optimality measure AUCPR used in cross-validation of the penalty parameters, significantly boosted the performance of the logistic regression method with the elastic net penalty, in terms of the AUCPR, the AUC, the variable selection property and binary classification performance. Meanwhile, the $elastic\ net^{AUC}$ did not give the highest AUC among the elastic nets, which seems to contradict the intuition that, a learning method optimized with respect to measure A, is supposed to

perform at least similarly to, if not better than, the same learning method optimized with respect to measure B, in terms of measure A. The $lasso^{AUCPR}$ achieved also the best variable selection property among the lassos, despite the fact that the ranking ability and Brier score of the lasso are hardly affected by the optimality measures used.

The learning from the credit card default data was so difficult that none of the regularization methods performed very well. Neither were the methods strongly affected by the optimality measure used in cross-validation of the penalty parameters. The $elastic\ net^{AUCPR}$ and $lasso^{AUCPR}$, however, provided respectively better results (e.g., the AUC, the AUCPR, the Agm, etc) than the other elastic nets and lassos, in the third decimal. This result might still be useful for investigators who are even interested in a small increase in the prediction performance, when no methods perform decently well for rather challenging imbalanced learning problems. Also, the Brier score produced by the penalized logistic regression methods was not very meaningful in comparing their sharpness. Therefore, we recommend that investigators report also the Brier score of the positive class, and the Brier score of the negative class, for class imbalanced data. For real fraud detection tasks where the ratio of class imbalance is unknown for new coming data, both the AUCPR and the AUC might be provided in order to draw more reliable conclusions on the models' prediction performance, since the baseline in the PR space is not invariant.

Overall, the results have shown that whether and how the optimality measure used in cross-validation of the penalty parameters, will affect the performance of each of the regularization methods, depends on many factors. The choice of measure of prediction performance used in the validation phase, and the choice of the measure used for achieving the optimal performance in the training phase, should be made with care. The two measures are not always consistent for highly imbalanced data, which is at least true for the AUC.

Part of the conclusions drawn from the simulation study and the illustration on the credit card default data are different, and the imbalanced learning from the credit card default data seems to be much more difficult than from the simulated data. In the simulation study, we could have also experimented with more complicated simulation settings, for instance, by increasing the overlapping among the classes to make the imbalanced learning problem more difficult. In such cases, more comprehensive conclusions could have been drawn with respect to the objective of our study.

In addition, the imputation method for the numerical data is relatively naive in the analysis of the credit card default data, which might contribute to the fact that the imbalanced learning from the simply imputed data is rather challenging. We could have also done the regression analysis on the relation between the outcome and covariates

with missing information, such that the missing numerical data are imputed in a more advanced way. In such cases, different results may be obtained.

As we noticed that, for severe imbalance, or very high-dimension, each of the penalized logistic regression methods, performed equally poorly, regardless of the measures used in cross-validation of the penalty parameters. For such highly imbalanced and very high-dimensional data, other possible approaches may be helpful in improving the prediction performance (e.g., the AUCPR) of the penalized logistic regression models. For instance, re-sampling techniques which are discussed in Section 3.5 may be implemented before the training phase. Also, for highly class imbalanced data with a small number of positive classes, data cleaning methods using Tomek links may be, hopefully, effective in producing better results of the AUCPR.

Appendices

Appendix A

Figures and tables

A.1 Methods

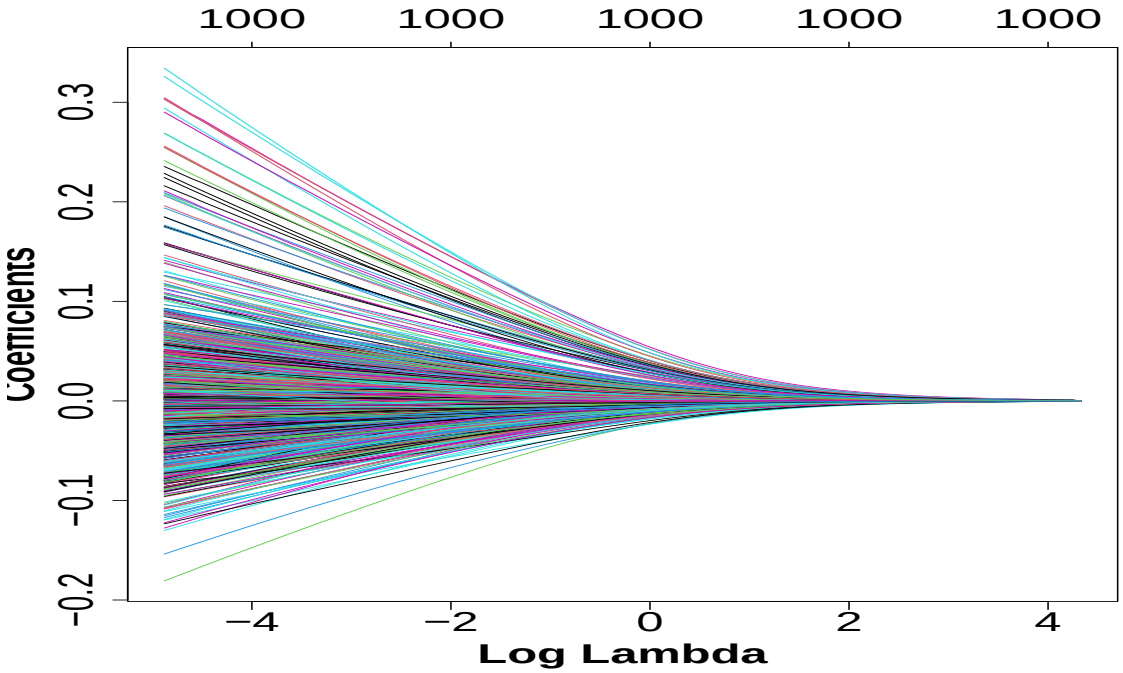


Figure A.1: Plot of how all the 1000 coefficients vary with the penalty parameter λ in the ridge regression. All coefficients are shrunk proportionally to very small values, but no coefficient will be exactly zero, as the λ increases from around 0.02 to 55.

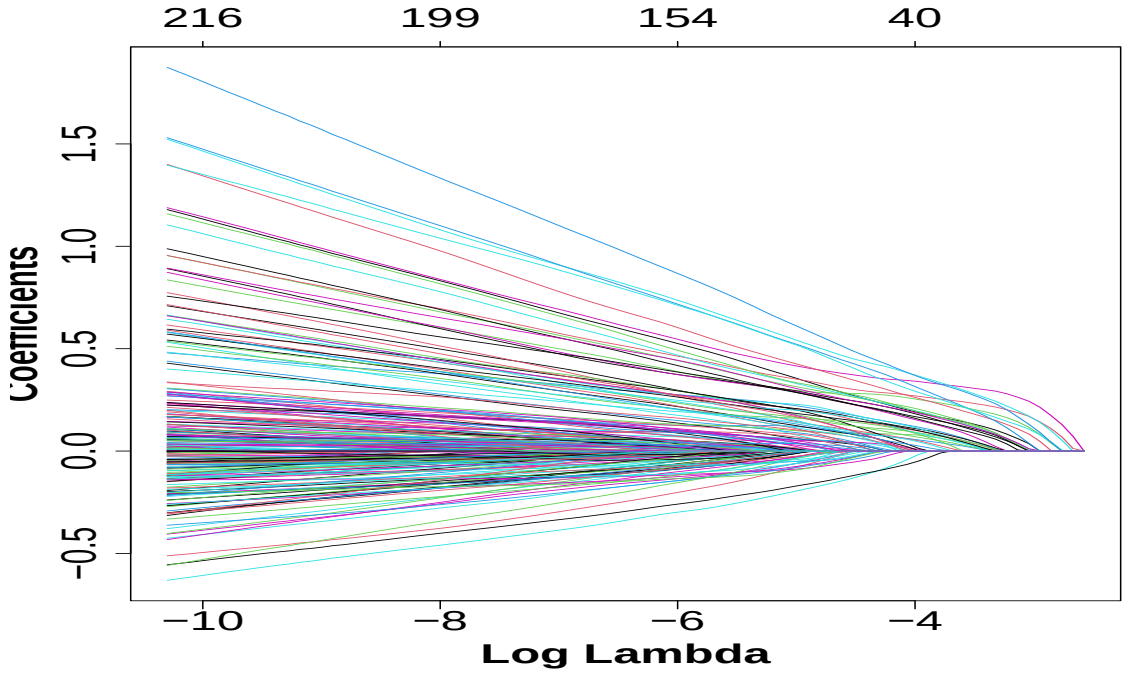


Figure A.2: Plot of how the lasso regression selects variables with different values of the penalty parameter λ.

A.2 Simulation study

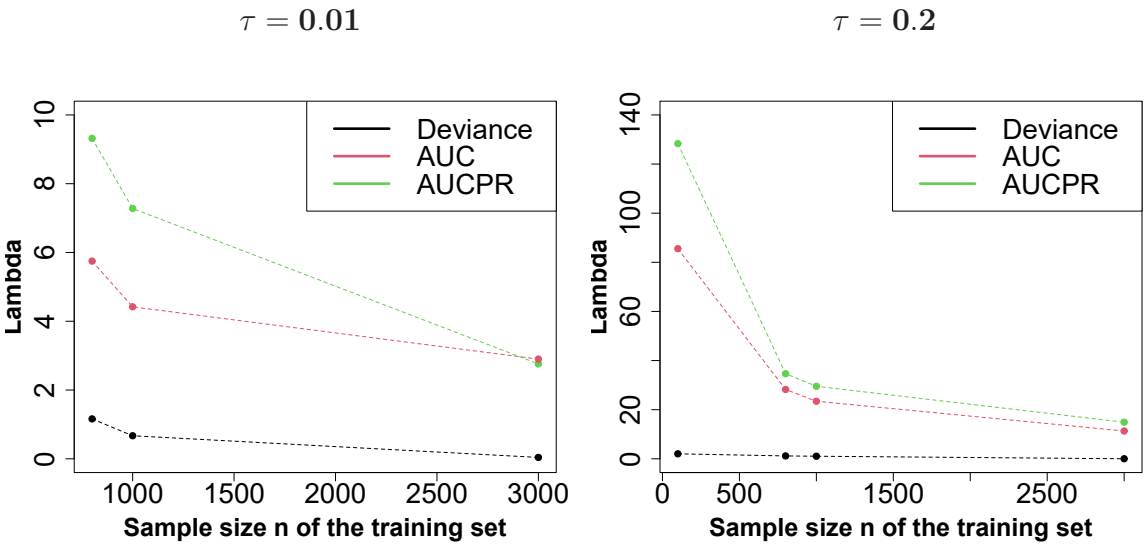


Figure A.3: Plots of average shrinkage parameter λ chosen by the three measures in the ridge regression.

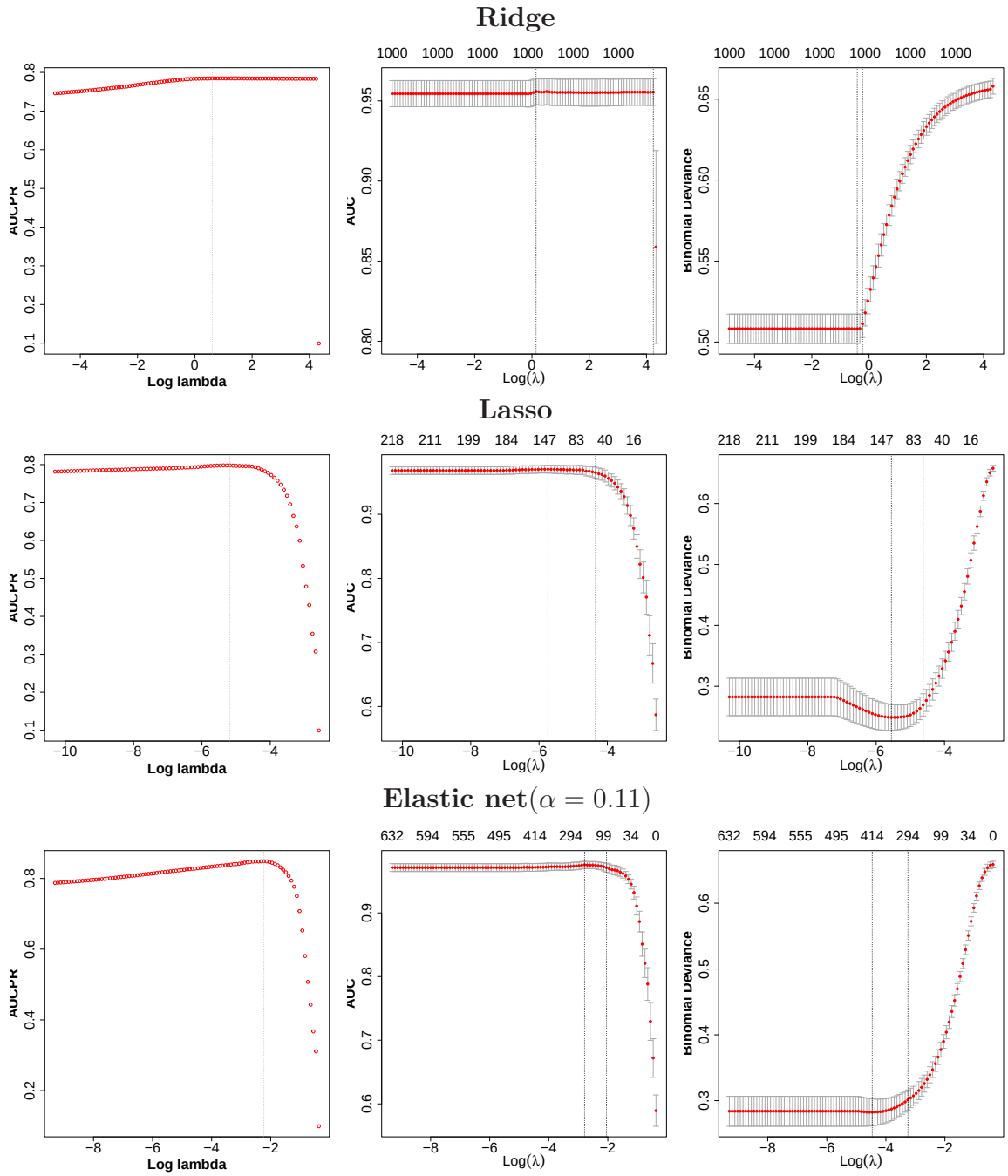


Figure A.4: Cross-validation plots of how the AUCPR *(left)*, AUC *(middle)* and deviance *(right)* vary with the penalty parameter λ in the ridge *(upper panel)*, lasso *(middle panel)* and elastic net *(bottom panel)* regression for $\tau = 0.1$ and $n = 1000$. In the ridge and lasso regression, each of the three measures chooses its own optimal λ value from a very wide range of λ where the AUCPR stays constant.

Table A.1: Average α in the elastic net regression. A smaller α means a smaller percentage of the L_1-norm penalty and a larger percentage of the L_2-norm penalty. For reference, the ridge regression is with $\alpha = 0$ and the lasso regression is with $\alpha = 1$.

		Elnet		
τ	n	Deviance	AUC	AUCPR
0.01	800	0.53	0.41	0.51
	1000	0.54	0.35	0.39
	3000	0.70	0.37	0.27
0.05	800	0.62	0.37	0.21
	1000	0.59	0.49	0.17
	3000	0.71	0.71	0.11
0.1	100	0.48	0.39	0.29
	800	0.52	0.53	0.14
	1000	0.56	0.54	0.11
	3000	0.79	0.85	0.10
0.2	100	0.51	0.29	0.29
	800	0.52	0.58	0.12
	1000	0.58	0.68	0.12
	3000	0.89	0.85	0.10

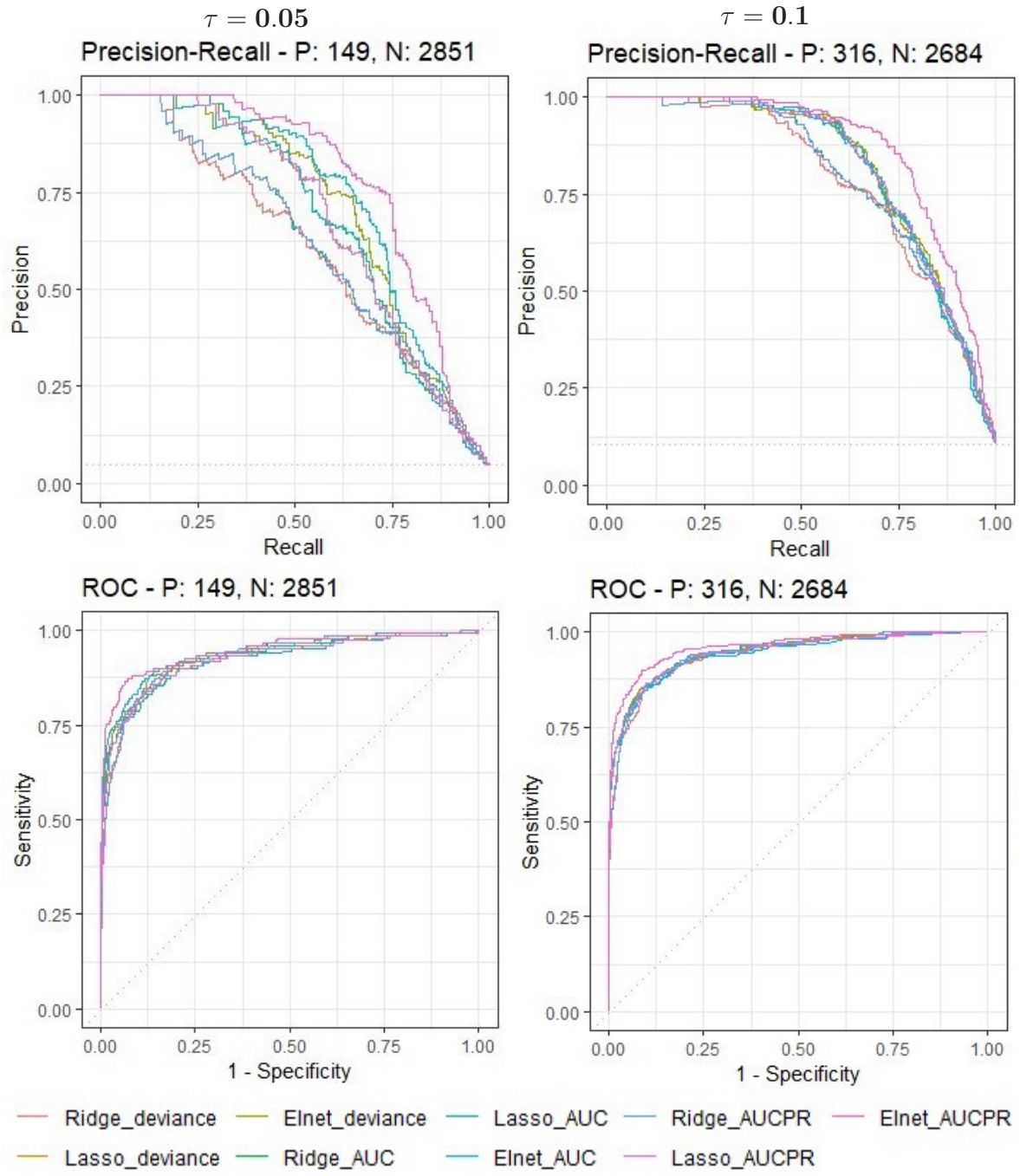


Figure A.5: PR curves *(upper panel)* and ROC curves *(bottom panel)* of all regularization methods. For each of the two ratios of class imbalance, all models are trained on the same 1000×1000 training set and evaluated on the same 3000×1000 test set.

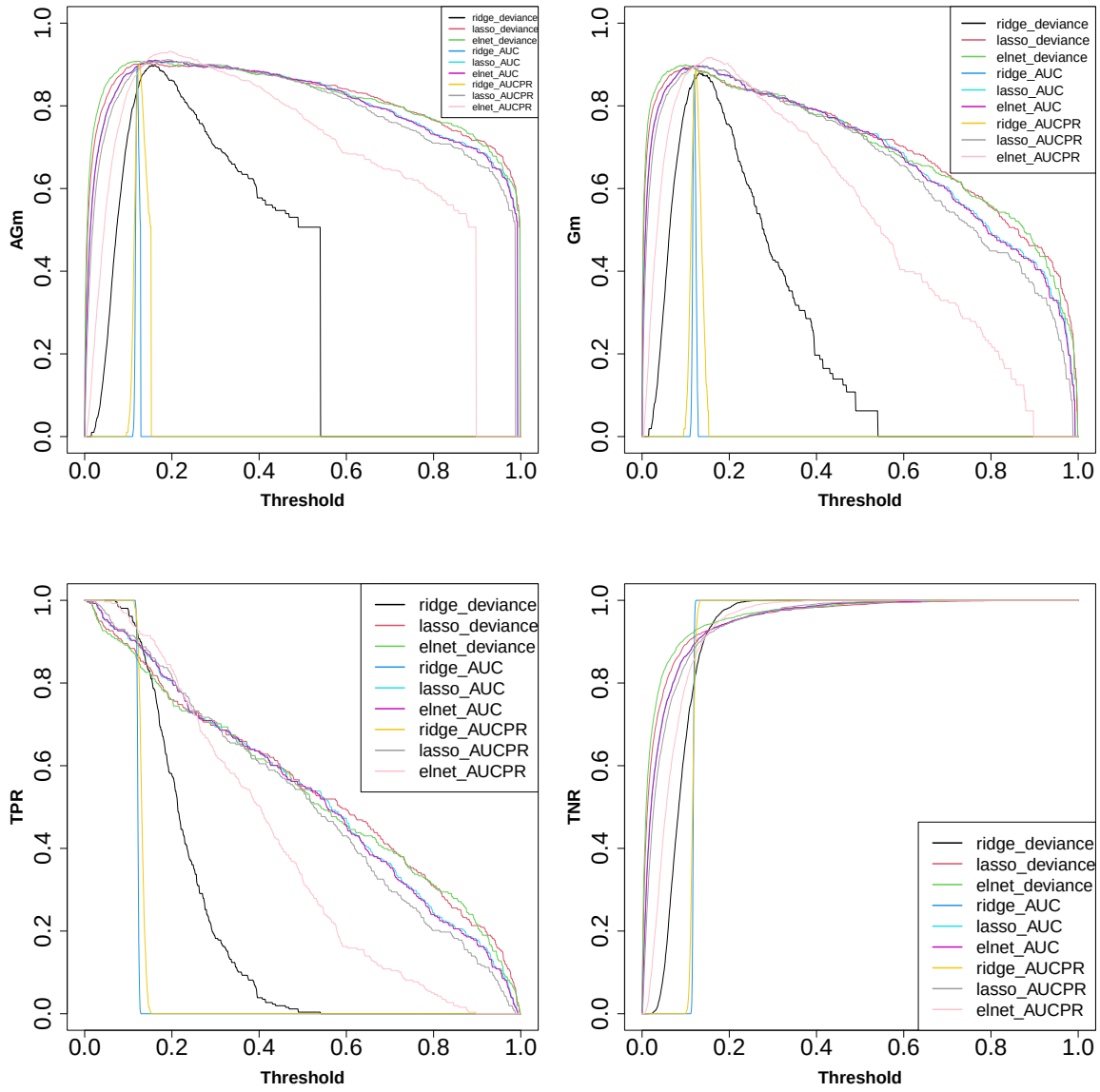


Figure A.6: The AGm, Gm, TPR and TNR curves of all the regularization methods from one simulation, which is run with $\tau = 0.1$ and $n = 1000$. The $elastic\ net^{AUCPR}$ is the best, and it achieves the highest AGm and Gm at the thresholds of 0.197 and 0.16, respectively.

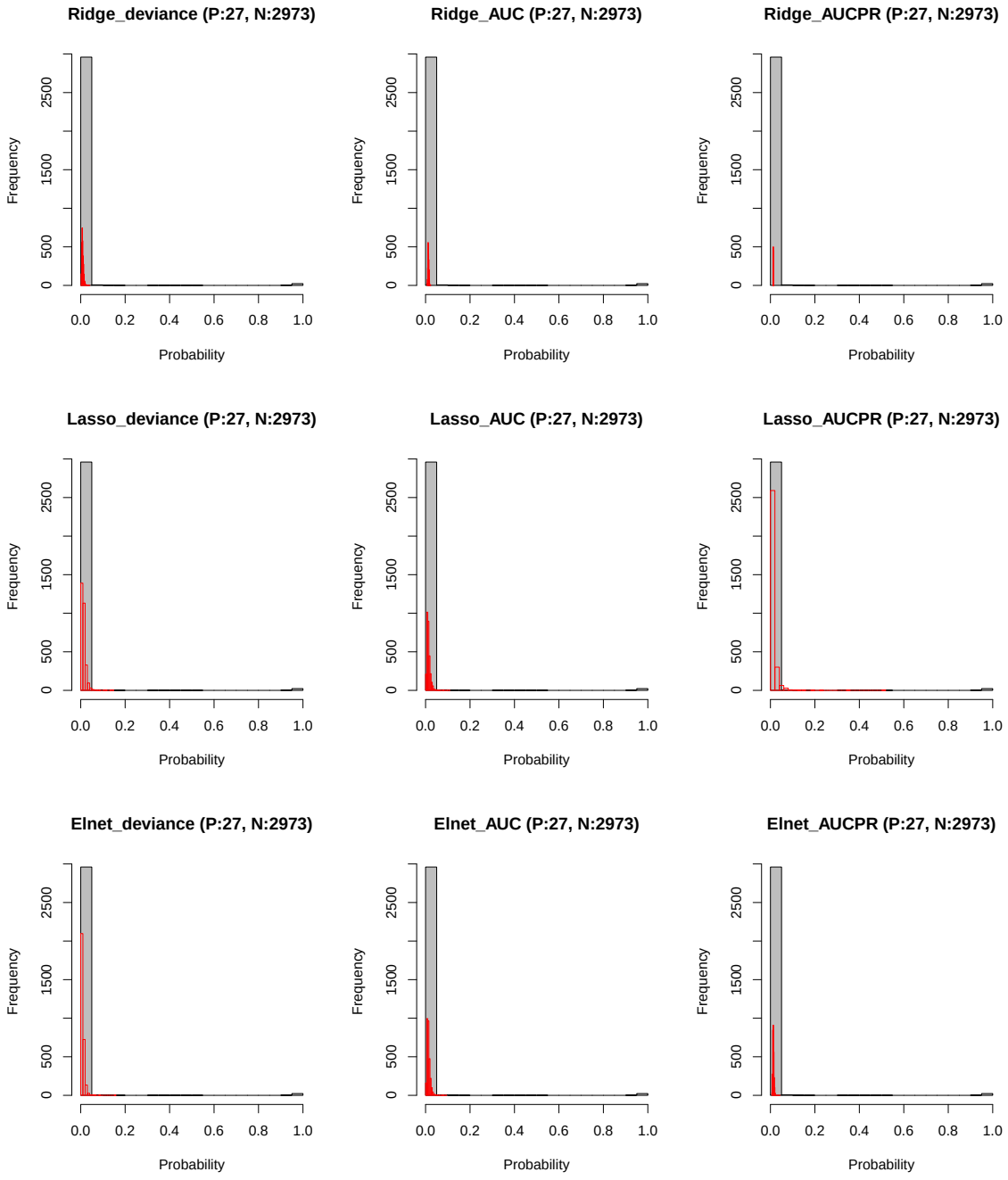
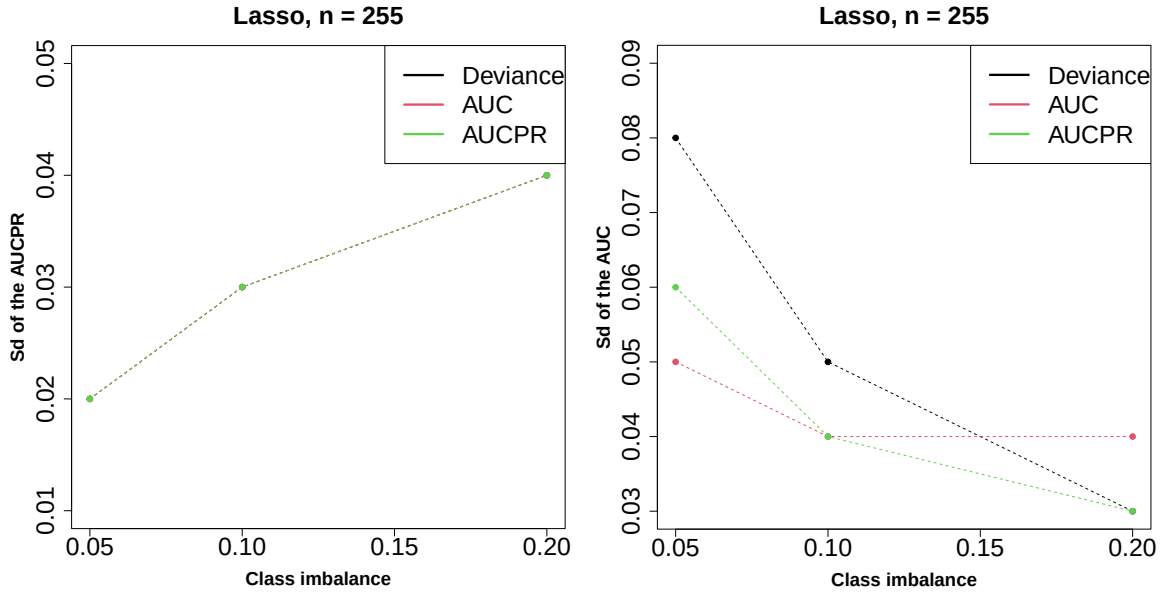


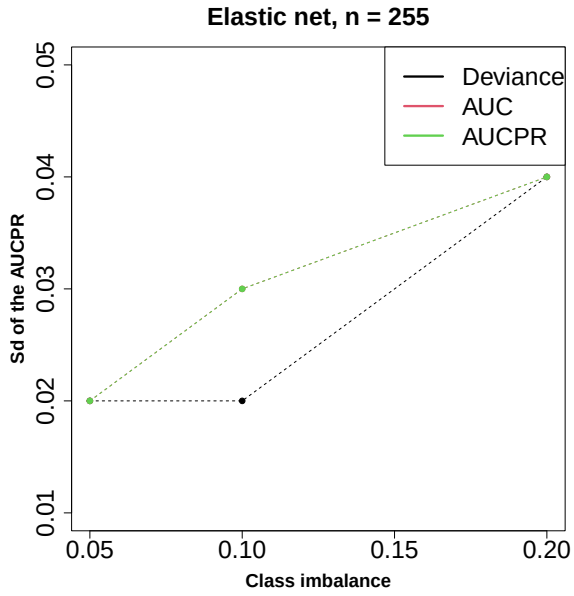
Figure A.7: Histogram plots of predicted event probabilities *(red)* and true event probabilities *(grey)*, for $\tau = 0.01$ and $n = 1000$.

A.3 Illustration on credit card default data

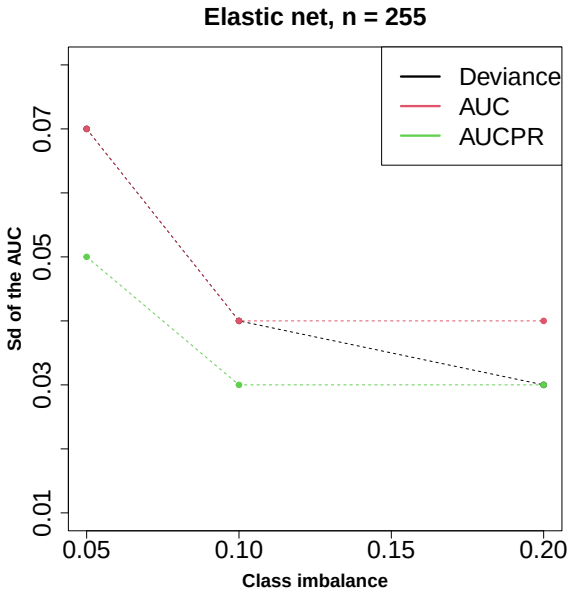


(a)

(b)



(c)



(d)

Figure A.8: Plots of the average standard deviation of the AUCPR and AUC in the lasso *(upper panel)* and elastic net *(bottom panel)* regression, respectively. In Figure A.8a, the lasso gives the same standard deviation of the AUCPR, regardless of the optimality measure used in cross-validation of the penalty parameter. In Figure A.8c, the *elastic net*^{AUCPR} and *elastic net*^{AUC} have the same standard deviation of the AUCPR.

$Ridge^{deviance}$

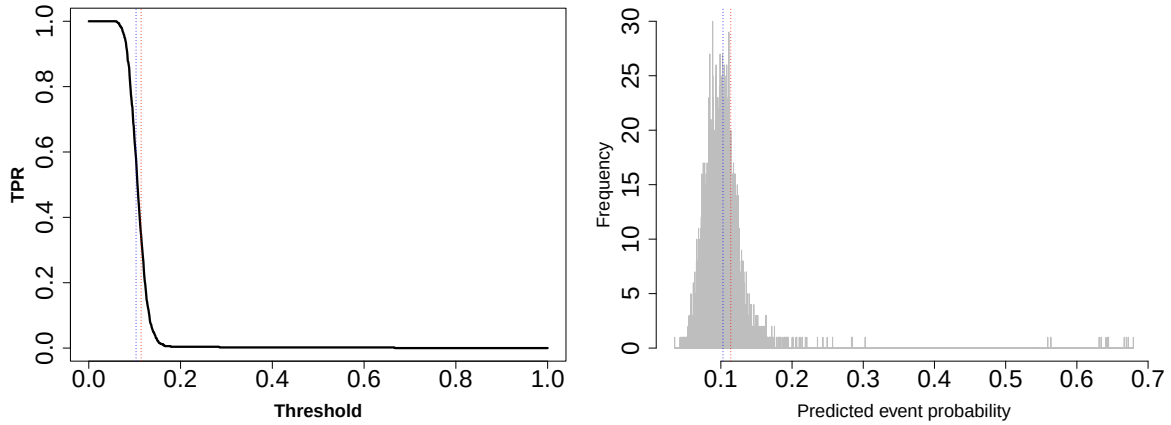


Figure A.9: The TPR curve over all the thresholds *(left)* and the histogram plot of the predicted event probabilities *(right)* given by $ridge^{deviance}$, for $\tau = 0.1$, $n = 255$, and $m = 5000$ (i.e., the number of cases in the test set). The training and test data are sampled from the credit card default data. The TPR curves of the other regularization methods follow a similar pattern, and thus not given. The blue line is the threshold value of 0.103 chosen by the maximal Gm, and the red line is the threshold value of 0.114 chosen by the maximal AGm.

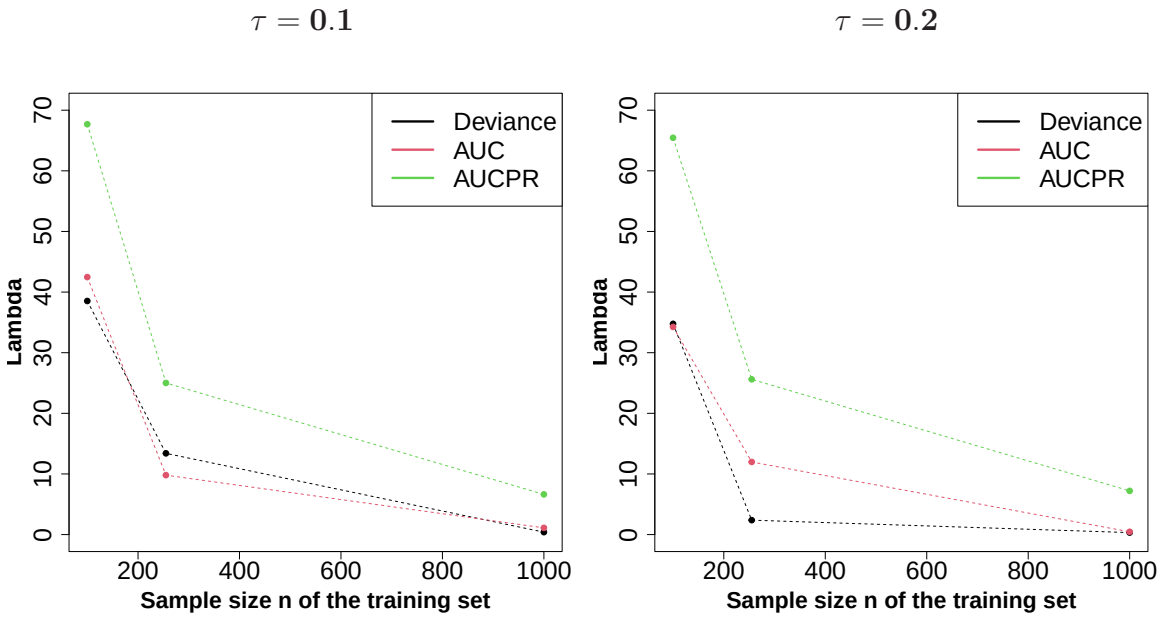


Figure A.10: Plots of average shrinkage parameter λ chosen by the three measures in the ridge regression.

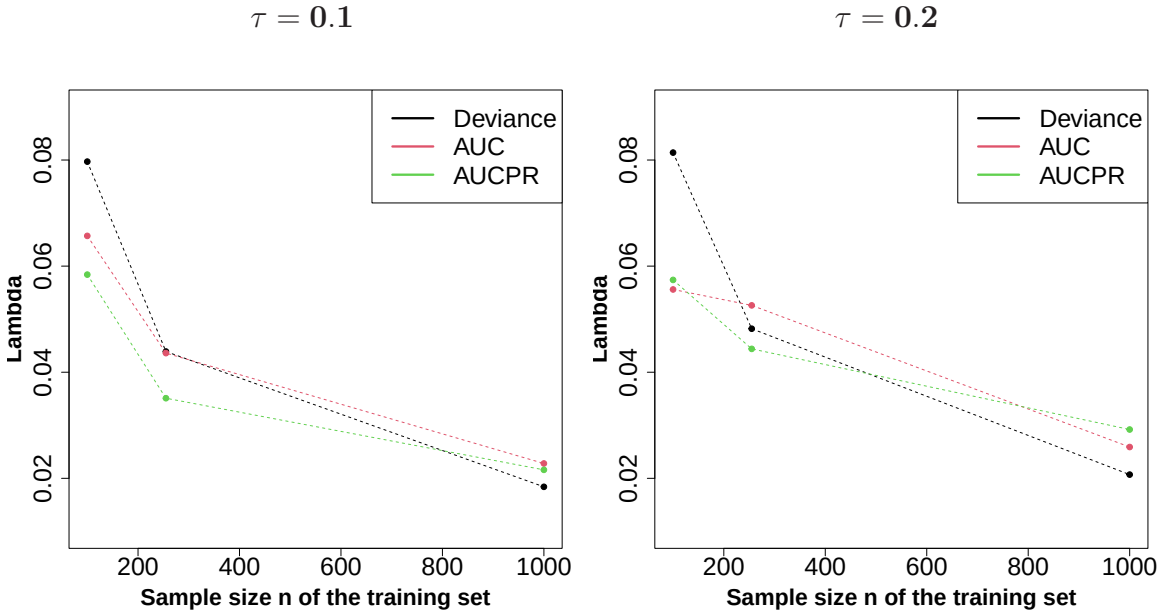


Figure A.11: Plots of average shrinkage parameter λ chosen by the three measures in the lasso regression.

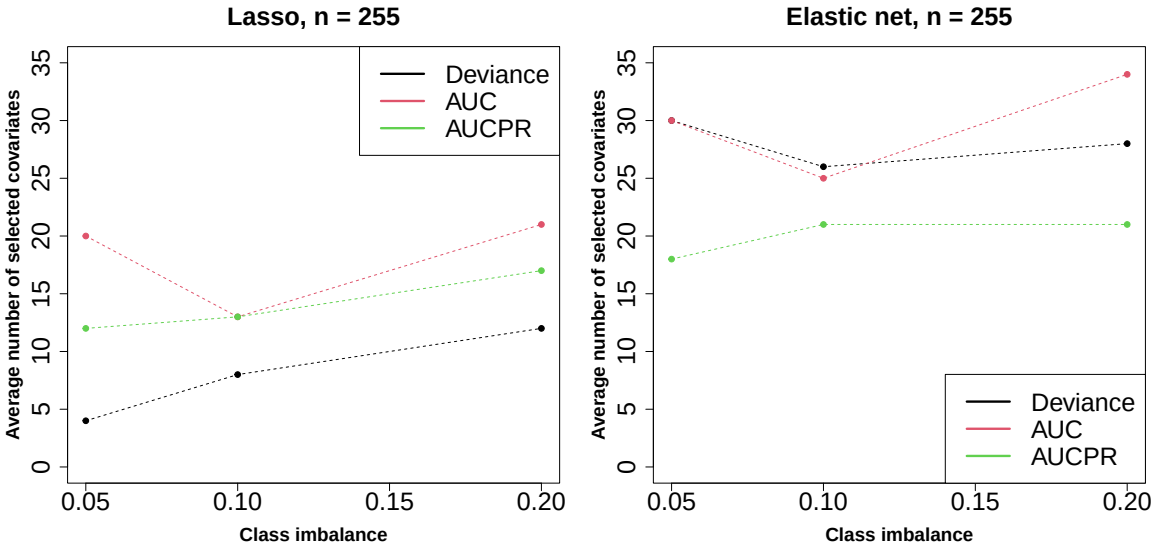


Figure A.12: Plots of average number of selected covariates in the lasso and elastic net regression, respectively. Here, the sample size $n = 255$, which is also the number p of covariates.

Appendix B

R-code

B.1 Data simulation

```r
library(MASS)
library(glmnet) # library to perform the ridge, lasso and elastic net regression
library(precrec) # library to compute the AUCPR and the AUC
library(caret) # library to compute the confusion matrix
library(doParallel) # library for parallel computing
library(rvinecopulib) # library to compute the true event probabilities

#Generate the correlation matrices
library(randcorr)
set.seed(12022022)
rho <- function(p){
  rho<- randcorr(p)
  if(sum(eigen(rho)$values<= 0)>0 ){
    cat(" Correlation matrix is not positive definite.")
  }else{
    return(rho)
  }
}
rho0 <- rho(1000);rho1<- rho(1000)  # Generate two pre-fixed different correlation matrices
write.table(rho0, file=sprintf("rho0"));
write.table(rho1, file=sprintf("rho1"));

rho0<-read.table(file=sprintf("rho0"))
rho1<-read.table(file=sprintf("rho1"))


#Define sim_dat for generating the data by the Gaussian copula approach
  sim_dat <- function(n,p,p.y,mu1,sigma1,alpha1,gamma1,pB1_1,pB2_1,ind.cov){
    y<-rbinom(n,size=1, prob = p.y)
    index0<-which(y==0)
```

```
n0<-length(index0)

index1<-which(y==1)
n1<-length(index1)

z0<-u0<-x0<-array(NA, c(n0,p))
z0<-mvrnorm(n0,rep(0,p),rho0)
u0<-pnorm(z0)

z1<-u1<-x1<-array(NA, c(n1,p))
z1<-mvrnorm(n1,rep(0,p),rho1)
u1<-pnorm(z1)

marg.types <- c(rep("norm",300),rep("gamma",300),
rep("bern_1",200),rep("bern_2",200))
if(p==1000){
  for(i in 1:p)
  {
    if(marg.types[i] == "norm")
    {
      x0[,i]<-qnorm(u0[,i],0,1)
      if(is.element(i,ind.cov))
      {
        x1[,i]<-qnorm(u1[,i],mu1,sigma1)
      }
      else
      {
        x1[,i]<-qnorm(u1[,i],0,1)
      }
    }
    else if(marg.types[i] == "gamma")
    {
      x0[,i]<-qgamma(u0[,i],2,2)
      if(is.element(i,ind.cov))
      {
        x1[,i]<-qgamma(u1[,i],alpha1,gamma1)
      }
      else
      {
        x1[,i]<-qgamma(u1[,i],2,2)
      }
    }
    else if(marg.types[i] == "bern_1")
    {
      x0[,i]<-1*(u0[,i]<0.1)
      if(is.element(i,ind.cov))
      {
        x1[,i]<-1*(u1[,i]<pB1_1)
      }
```

```
          else
          {
            x1[,i]<-1*(u1[,i]<0.1)
          }
        }
        else if(marg.types[i] == "bern_2")
        {
          x0[,i]<-1*(u0[,i]<0.5)
          if(is.element(i,ind.cov))
          {
            x1[,i]<-1*(u1[,i]<pB2_1)
          }
          else
          {
            x1[,i]<-1*(u1[,i]<0.5)
          }
        }
      }

    }else {cat("The dimension of predicitors p is not equal to 1000.")}
    dat0<-cbind(rep(0,n0),x0)
    dat1<-cbind(rep(1,n1),x1)
    dat<-rbind(dat1,dat0)
    rm(u0,z0,u1,z1)
    gc()
    return(dat)
  }
```

B.2 Stratified k-fold cross-validation

```
scv <- function(y,k){              #y is the target set
  index0<-which(y==0)
  n0<-length(index0)

  index1<-which(y==1)
  n1<-length(index1)

  n<-n0+n1
  index<-  numeric(n)
  if ((n0%%k) == 0){
    index[index0]<-sample(rep(1:k,times = n0%/%k))
  }else if((n0%%k) > 0){
    index[index0]<-sample(c(rep(1:k,times=n0%/%k),1:(n0%%k)))
  }
  if ((n1%%k) == 0){
    index[index1]<-sample(rep(1:k,times = n1%/%k))
  }else if((n1%%k) > 0){   #n1%%k returns the remainder
    index[index1]<-sample(c(rep(1:k,times=n1%/%k),1:(n1%%k)))
```

```
    }
    return(index)
  }
```

B.3 Computing the true event probabilities

The code in B.3 is provided by my supervisor Ingrid Hobæk Haff.

```
   #Computing the true event probabilities
  compute.prob <- function(y,x,p.y,marg.types,marg.par.0,
  marg.par.1,corr.mat.0,corr.mat.1,var.types,
  vine.0=NULL,vine.1=NULL,ind.cov=NULL)
  {
    d <- ncol(x)
    n <- nrow(x)

    if(length(ind.cov) == 0)
    {
      ind.cov <- 1:d
    }

    if(length(vine.0) == 0)
    {
      vine.comp.0 <- construct.vine(d,corr.mat.0,var.types)
      vine.comp.1 <- construct.vine(d,corr.mat.1,var.types)

      vine.0 <- vinecop_dist(vine.comp.0$pair.copulas,
      vine.comp.0$vine.structure,var.types)
      vine.1 <- vinecop_dist(vine.comp.1$pair.copulas,
      vine.comp.1$vine.structure,var.types)
    }


    log.prob.x.0 <- compute.prob.x(x,vine.0,marg.types,marg.par.0,var.types,ind.cov)
    log.prob.x.1 <- compute.prob.x(x,vine.1,marg.types,marg.par.1,var.types,ind.cov)

    prob <- exp(log.prob.x.1-log.prob.x.0)/(exp(log.prob.x.1-log.prob.x.0)+(1-p.y)/p.y)

    prob
  }

  compute.prob.x <- function(x,vine,marg.types,marg.par,var.types,ind.cov)
  {
    d <- ncol(x)
    u <- F.all(x,marg.types,marg.par,var.types)
    prob <- dvinecop(u,vine)
    log.prob <- log(prob)
```

```
  for(i in 1:d)
  {
    if(is.element(i,ind.cov))
    {
      if(marg.types[i] == "norm")
      {
        log.prob <- log.prob+dnorm(x[,i],
        marg.par[[i]][1],marg.par[[i]][2],log=TRUE)
      }
      else if(marg.types[i] == "gamma")
      {
        log.prob <- log.prob+dgamma(x[,i],
        marg.par[[i]][1],marg.par[[i]][2],log=TRUE)
      }
      else if(marg.types[i] == "lnorm")
      {
        log.prob <- log.prob*dlnorm(x[,i],
        marg.par[[i]][1],marg.par[[i]][2],log=TRUE)
      }
      else if(marg.types[i] == "t")
      {
        log.prob <- log.prob+dt(x[,i],marg.par[[i]][1],log=TRUE)
      }
      else if(marg.types[i] == "bern")
      {
        log.prob <- log.prob+x[,i]*
        log(marg.par[[i]][1])+(1-x[,i])*log(1-marg.par[[i]][1])
      }
    }
  }

  log.prob
}

construct.vine <- function(d,corr.mat,var.types)
{
  vine.structure <- dvine_structure(1:d)
  pair.copulas <- list(length = d-1)
  pair.copulas[[1]] <- list()
  for(i in 1:(d-1))
  {
    pair.copulas[[1]][[i]] <- bicop_dist("gaussian",0,
    corr.mat[i,i+1],var_types=var.types[c(i,i+1)])
  }
  for(i in 2:(d-1))
  {
    pair.copulas[[i]] <- list()
    for(j in 1:(d-i))
    {
```

```
      p.mat <- solve(corr.mat[j:(j+i),j:(j+i)])
      part.corr.ij <- -p.mat[1,1+i]/sqrt(p.mat[1,1]*p.mat[1+i,1+i])
      pair.copulas[[i]][[j]] <- bicop_dist("gaussian",0,
      part.corr.ij,var_types=var.types[c(j,j+i)])
    }
  }

  list(vine.structure=vine.structure,pair.copulas=pair.copulas)
}

construct.vine.level <- function(d,corr.mat,var.types,levels)
{
  pair.copulas <- list(length = length(levels))
  for(i in 1:length(levels))
  {
    pair.copulas[[i]] <- list(length = d-levels[i])
    if(levels[i] == 1)
    {
      for(j in 1:(d-1))
      {
        pair.copulas[[i]][[j]] <- bicop_dist("gaussian",0,
        corr.mat[j,j+1],var_types=var.types[c(j,j+1)])
      }
    }
    else
    {
      for(j in 1:(d-levels[i]))
      {
        p.mat <- solve(corr.mat[j:(j+levels[i]),j:(j+levels[i])])
        part.corr.ij <- -p.mat[1,1+levels[i]]/
        sqrt(p.mat[1,1]*p.mat[1+levels[i],1+levels[i]])
        pair.copulas[[i]][[j]] <- bicop_dist("gaussian",0,part.corr.ij,
        var_types=var.types[c(j,j+levels[i])])
      }
    }
  }

  pair.copulas
}

F.all <- function(x,marg.types,marg.par,var.types)
{
  d <- ncol(x)

  u.x <- x
  u.x.discr <- c()
  for(i in 1:d)
  {
    if(var.types[i] == "c")
```

```
    {
      if(marg.types[i] == "norm")
      {
        u.x[,i] <- pnorm(x[,i],marg.par[[i]][1],marg.par[[i]][2])
      }
      else if(marg.types[i] == "gamma")
      {
        u.x[,i] <- pgamma(x[,i],marg.par[[i]][1],marg.par[[i]][2])
      }
      else if(marg.types[i] == "lnorm")
      {
        u.x[,i] <- plnorm(x[,i],marg.par[[i]][1],marg.par[[i]][2])
      }
      else if(marg.types[i] == "t")
      {
        u.x[,i] <- pt(x[,i],marg.par[[i]][1])
      }
    }
    else
    {
      u.x[,i] <- 1-marg.par[[i]][1]*as.numeric(x[,i] == 0)
      u.x.discr <- cbind(u.x.discr,
      (1-marg.par[[i]][1])*as.numeric(x[,i] == 1))
    }
  }
  u.x <- cbind(u.x,u.x.discr)

  as.matrix(u.x,nrow=nrow(x))
}
```

B.4 Evaluation metrics

```
metrics<-function(model, x, y, p.y,p.y.1){

  p.y.1_1<-p.y.1[which(y==1)]
  p.y.1_0<-p.y.1[which(y==0)]

  y.predicted<-predict(model, newx = x, type = 'response')
  p1<-length(which(y==1));p2<-length(which(y==0))
  y.predicted_1<-y.predicted[which(y==1)]
  y.predicted_0<-y.predicted[which(y==0)]

  find<-find_gm(y.predicted,y,p.y) #Threshold based metrics
  Gm<-find[2] #Maximal Gm of TPR and TNR
  t_gm<-find[1]  #Threshold which maximizes the Gm
  Gm_sensi<-find[7] #TPR
  Gm_spec<-find[8] #TNR

```

```
    Agm<- find[6]
    t_Agm<-find[5]  #Threshold which maximizes the AGm
    Agm_sensi<-find[11] #TPR
    Agm_spec<-find[12] #TNR

    curves<-evalmod(scores=y.predicted, labels=y);aucs<-auc(curves);
    roc_auc<-aucs[1,4]  #The AUC
    pr_auc<-aucs[2,4]   #The AUCPR


    bs<- mean((y.predicted-y)^2) # Brier score
    bs_1<-(sum((y.predicted_1-1)^2))/p1
    bs_0<-(sum((y.predicted_0)^2))/p2

    bias<-mean(y.predicted-p.y.1)  #Bias
    bias_1<-(sum(y.predicted_1-p.y.1_1))/p1
    bias_0<-(sum(y.predicted_0-p.y.1_0))/p2

    rmse<-sqrt(mean((y.predicted-p.y.1)^2)) #Rmse

    ranking_score<-ranking(y.predicted,p.y.1,y) #The actual ranking score

    return(rbind(pr_auc,roc_auc,ranking_score,
    Agm,Agm_sensi,Agm_spec, Gm,Gm_sensi,Gm_spec,
     bs,bs_1,bs_0,bias,bias_1,bias_0,rmse))
  }
```

B.5 Exporting the results of the simulation study

```
loss.types <- c("deviance","AUC","AUCPR")
method.types <- c("ridge","lasso","elnet")
var.types <- c(rep("c",600),rep("d",400))
marg.types <- c(rep("norm",300),rep("gamma",300),rep("bern",200),rep("bern",200))
ind.cov <- c(1:30,1:30+300,1:20+600,1:20+800) #indices of significant covariates
marg.par.0 <- marg.par.1 <- list()
for(i in 1:length(marg.types))
{
  if(i <= 300)
  {
    marg.par.0[[i]] <- c(0,1)
    if(is.element(i,ind.cov))# Covariate i affects Y
    {
      marg.par.1[[i]] <- c(0.6,1.5)
    }
    else# Covariate i does not affect Y
    {
      marg.par.1[[i]] <- c(0,1)
    }
```

```
  }
  else if((i > 300)&(i <= 600))
  {
    marg.par.0[[i]] <- c(2,2)
    if(is.element(i,ind.cov))# Covariate i affects Y
    {
      marg.par.1[[i]] <- c(2.05,2.01)
    }
    else# Covariate i does not affect Y
    {
      marg.par.1[[i]] <- c(2,2)
    }
  }
  else if((i > 600)&(i <= 800))
  {
    marg.par.0[[i]] <- 0.1
    if(is.element(i,ind.cov))# Covariate i affects Y
    {
      marg.par.1[[i]] <- 0.2
    }
    else# Covariate i does not affect Y
    {
      marg.par.1[[i]] <- 0.1
    }
  }
  else
  {
    marg.par.0[[i]] <- 0.5
    if(is.element(i,ind.cov))# Covariate i affects Y
    {
      marg.par.1[[i]] <- 0.4
    }
    else# Covariate i does not affect Y
    {
      marg.par.1[[i]] <- 0.5
    }
  }
}
rho0<-read.table(file=sprintf("rho_0"))
rho1<-read.table(file=sprintf("rho_1"))

load("/home/shuijinl/vine.0.RData")
load("/home/shuijinl/vine.1.RData")

 result_imbalanced<-function(n,m,p,p.y,k,mu1,sigma1,
 alpha1,gamma1,pB1_1,pB2_1,loss.types,method.types){
    p.y<-p.y #The proportion of positive cases in the data

    #Generate the training set
```

```
69    train<-sim_dat(n,p,p.y,mu1,sigma1,alpha1,gamma1,pB1_1,pB2_1,ind.cov)
70    #Generate the test set
71    test<-sim_dat(m,p,p.y,mu1,sigma1,alpha1,gamma1,pB1_1,pB2_1,ind.cov)
72    x.train<-train[,-1]
73    y.train<-train[,1]
74    x.test<-test[,-1]
75    y.test<-test[,1]
76
77    #Compute the true event probabilities of the test set
78    p.y.1 <- compute.prob(y.test,x.test,p.y,marg.types,
79    marg.par.0,marg.par.1,corr.mat.0=rho0,
80    corr.mat.1=rho1,var.types,vine.0,vine.1,ind.cov)
81    #Standardization for pure prediction
82    x.mean <- apply(x.train, 2, mean)
83    x.sd <- apply(x.train, 2, sd)
84    x.train <- scale(x.train, center = TRUE, scale = TRUE) #default: scale(x, center = TRUE, sca
85
86    #Using the train mean and sd to standardize the test set
87    x.test <- sapply(1:ncol(x.test),
88                     function(i, x.test, x.mean, x.sd) (x.test[, i] - x.mean[i])/x.sd[i],
89                     x.test = x.test, x.mean = x.mean, x.sd = x.sd)
90
91    index <- scv(y.train,k) #Stratified k-folds cross-validation
92    registerDoParallel(2)
93    for (a in loss.types){
94      loss<-a
95      for (b in method.types){
96        method<-b
97        if (loss == 'deviance'){
98          if (method == 'ridge'){
99
100           lambda.ridge <-cv.glmnet(x.train, y.train,
101           family = 'binomial', alpha = 0, foldid = index,
102           parallel = TRUE)$lambda.min
103           mod.ridge <-glmnet(x.train, y.train,
104           lambda = lambda.ridge, family = 'binomial', alpha = 0)
105           error<-metrics(mod.ridge,x.test,y.test,p.y,p.y.1)
106           result1<-rbind(lambda.ridge,error)
107         }else if(method == 'lasso'){
108
109           lambda.lasso <-cv.glmnet(x.train, y.train,
110           family = 'binomial', alpha = 1, foldid = index,
111           parallel = TRUE)$lambda.min
112           mod.lasso <-glmnet(x.train, y.train,
113           lambda = lambda.lasso, family = 'binomial', alpha = 1)
114           error<-metrics(mod.lasso,x.test,y.test,p.y,p.y.1)
115           v_s<-vs(mod.lasso,ind.cov) #Variable selection property
116           result2<-rbind(lambda.lasso,v_s,error)
117         }else if(method == 'elnet'){
```

```
118
119           findLambda <- function(x, y, k, a)
120           {
121             tmp <- cv.glmnet(x, y, family = 'binomial',
122             alpha = a, foldid = k, parallel = TRUE)
123             which.min(tmp$cvm)
124             c(min(tmp$cvm), tmp$lambda.min)
125           }
126           cv <- sapply(seq(0.1, 0.9, by = 0.1),
127           findLambda, x = x.train, y = y.train, k = index)
128           par <- list(alpha = NULL, lambda = NULL, choice = NULL)
129           par$choice <- which.min(cv[1, ])
130           alpha.elnet<-par$alpha <- seq(0.1, 0.9, by = 0.1)[par$choice]
131           lambda.elnet<-par$lambda <- cv[2, par$choice]
132           mod.elnet <- glmnet(x.train, y.train,
133           family = 'binomial', alpha = par$alpha,
134           lambda = par$lambda)
135           error<-metrics(mod.elnet,x.test,y.test,p.y,p.y.1)
136           v_s<-vs(mod.elnet,ind.cov)
137           result3<-rbind(alpha.elnet,lambda.elnet,v_s,error)
138         }
139       }else if(loss == 'AUC'){
140         if (method == 'ridge'){
141
142           lambda.ridge <-cv.glmnet(x.train, y.train,
143           family = 'binomial', alpha = 0,
144           type.measure = 'auc', foldid = index, parallel = TRUE)$lambda.min
145           mod.ridge <-glmnet(x.train, y.train,
146           lambda = lambda.ridge, family = 'binomial', alpha = 0)
147           error<-metrics(mod.ridge,x.test,y.test,p.y,p.y.1)
148           result4<-rbind(lambda.ridge,error)
149         }else if(method == 'lasso'){
150
151           lambda.lasso <-cv.glmnet(x.train, y.train,
152           family = 'binomial', alpha = 1,
153           type.measure = 'auc', foldid = index, parallel = TRUE)$lambda.min
154           mod.lasso <-glmnet(x.train, y.train,
155           lambda = lambda.lasso, family = 'binomial', alpha = 1)
156           error<-metrics(mod.lasso,x.test,y.test,p.y,p.y.1)
157           v_s<-vs(mod.lasso,ind.cov)
158           result5<-rbind(lambda.lasso,v_s,error)
159
160         }else if(method == 'elnet'){
161
162           findLambda <- function(x, y, k, a)
163           {
164             tmp <- cv.glmnet(x, y, family = 'binomial',
165             alpha = a, type.measure = 'auc',
166             foldid = k, parallel = TRUE)
```

```
            which.min(tmp$cvm)
            c(min(tmp$cvm), tmp$lambda.min)
          }
          cv <- sapply(seq(0.1, 0.9, by = 0.1),
          findLambda, x = x.train, y = y.train, k = index)
          par <- list(alpha = NULL, lambda = NULL, choice = NULL)
          par$choice <- which.min(cv[1, ])
          alpha.elnet<-par$alpha <- seq(0.1, 0.9, by = 0.1)[par$choice]
          lambda.elnet<-par$lambda <- cv[2, par$choice]
          mod.elnet <- glmnet(x.train, y.train,
          family = 'binomial', alpha = par$alpha,
          lambda = par$lambda)
          error<-metrics(mod.elnet,x.test,y.test,p.y,p.y.1)

          v_s<-vs(mod.elnet,ind.cov)
          result6<-rbind(alpha.elnet,lambda.elnet,v_s,error)

        }
      } else if(loss == 'AUCPR'){
        if (method == 'ridge'){
          lambda.ridge<-get_lambda(k,20,y.train,x.train,0)[1]
          mod.ridge <-glmnet(x.train, y.train,
          lambda = lambda.ridge, family = 'binomial', alpha = 0)
          error<-metrics(mod.ridge,x.test,y.test,p.y,p.y.1)
          result7<-rbind(lambda.ridge,error)
        }else if(method == 'lasso'){
          lambda.lasso<-get_lambda(k,20,y.train,x.train,1)[1]
          mod.lasso <-glmnet(x.train, y.train,
          lambda = lambda.lasso, family = 'binomial', alpha = 1)
          error<-metrics(mod.lasso,x.test,y.test,p.y,p.y.1)
          v_s<-vs(mod.lasso,ind.cov)
          result8<-rbind(lambda.lasso,v_s,error)


        }else if(method == 'elnet'){
          alpha<- seq(0.1, 0.9, by = 0.1)
          z<-length(alpha)
          lam<-pr_a<-numeric(z)
          for (i in 1:z){
            p_max<-get_lambda(k,20,y.train,x.train,alpha[i])
            lam[i]<-p_max[1]
            pr_a[i]<-p_max[2]
          }
          alpha.elnet<-alpha[which.max(pr_a)]
          lambda.elnet<-lam[which.max(pr_a)]
          mod.elnet<-glmnet(x.train, y.train,
          family = 'binomial', alpha = alpha.elnet, lambda = lambda.elnet)
          error<-metrics(mod.elnet,x.test,y.test,p.y,p.y.1)
          v_s<-vs(mod.elnet,ind.cov)
```

```
            result9<-rbind(alpha.elnet,lambda.elnet,v_s,error)

          }
        }
      }
    }


    rbind(result1,result4,result7,result2,
    result5,result8,result3,result6,result9)

  }

  cores<-detectCores()-1
  myCluster <- makeCluster(cores)
  registerDoParallel(myCluster)

  er<-foreach(i=1:q,.combine='cbind',
  .packages=c('MASS','glmnet','precrec','caret',
  'rvinecopulib','doParallel'),.verbose=TRUE,
  .errorhandling="remove")%dopar%{
    res<-result_imbalanced(n,3000,1000,p.y,k,
    0.6,1.5,2.05,2.01,0.2,0.4,loss.types,method.types)
    res
  }
  stopCluster(myCluster)
  ave_err<-rowMeans(er, na.rm = T) #Average results
  err<-cbind(ave_err,er)
  err<-as.data.frame(err)
  write.table(err, file=sprintf
  ("imbalance_%.2f_%.0fx%.0f_%.0f_folds",p.y,n,1000,k))


}
```

Bibliography

Agresti, A. (2015). *Foundations of linear and generalized linear models*. John Wiley & Sons, pp. 132–133.

Arlot, S. and Celisse, A. (2010). "A survey of cross-validation procedures for model selection". In: *Statistics surveys* vol. 4, pp. 40–79.

Batista, G. E. et al. (2004). "A study of the behavior of several methods for balancing machine learning training data". In: *ACM SIGKDD explorations newsletter* vol. 6, no. 1, pp. 20–29.

Batuwita, R. and Palade, V. (2009). "A New Performance Measure for Class Imbalance Learning. Application to Bioinformatics Problems". In: *2009 International Conference on Machine Learning and Applications*, pp. 545–550.

Benedetti, R. (2010). "Scoring rules for forecast verification". In: *Monthly Weather Review* vol. 138, no. 1, pp. 203–211.

Berrar, D. (2019). "Performance Measures for Binary Classification". eng. In: *Encyclopedia of Bioinformatics and Computational Biology*. Elsevier Inc, pp. 546–560.

Blagus, R. and Lusa, L. (2010). "Class prediction for high-dimensional class-imbalanced data". In: *BMC bioinformatics* vol. 11, no. 1, pp. 1–17.

Bolton, R. J. and Hand, D. J. (2002). "Statistical fraud detection: A review". In: *Statistical science* vol. 17, no. 3, pp. 235–255.

Boyd, K. et al. (2012). "Unachievable region in precision-recall space and its effect on empirical evaluation". In: *Proceedings of the... International Conference on Machine Learning. International Conference on Machine Learning*, p. 349.

Bradley, A. P. (1997). "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern recognition* vol. 30, no. 7, pp. 1145–1159.

Breiman, L. and Spector, P. (1992). "Submodel selection and evaluation in regression. The X-random case". In: *International statistical review/revue internationale de Statistique*, pp. 291–319.

Brier, G. W. (1950). "Verification of Forecasts Expressed in Terms of Probability". In: *Monthly weather review*, pp. 1–3.

Burez, J. and Van den Poel, D. (2009). "Handling class imbalance in customer churn prediction". In: *Expert Systems with Applications*, pp. 4626–4636.

Caruana, R. (2000). "Learning from imbalanced data: Rank metrics and extra tasks". In: *Proc. Am. Assoc. for Artificial Intelligence (AAAI) Conf*, pp. 51–57.

Chai, T. and Draxler, R. R. (2014). eng. In: *Geoscientific model development* vol. 7, no. 3, pp. 1247–1250.

Chawla, N. V. et al. (2002). "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* vol. 16, pp. 321–357.

— (2004). "Editorial: special issue on learning from imbalanced data sets". eng. In: *SIGKDD explorations* vol. 6, no. 1, pp. 1–6.

Daskalaki, S. et al. (2006). "Evaluation of classifiers for an uneven class distribution problem". In: *Applied artificial intelligence*, pp. 381–417.

Davis, J. and Goadrich, M. (2006). "The relationship between Precision-Recall and ROC curves". In: *Proceedings of the 23rd international conference on Machine learning*, pp. 233–240.

Efron, B. and Hastie, T. (2021). *Computer Age Statistical Inference, Student Edition: Algorithms, Evidence, and Data Science*. Vol. 6. Cambridge University Press, p. 41.

Estabrooks, A. et al. (2004). "A multiple resampling method for learning from imbalanced data sets". In: *Computational intelligence* vol. 20, no. 1, pp. 18–36.

Fawcett, T. (2006). "An introduction to ROC analysis". In: *Pattern recognition letters* vol. 27, no. 8, pp. 861–874.

Galar, M. et al. (2012). "A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches". eng. In: *IEEE transactions on systems, man and cybernetics. Part C, Applications and reviews* vol. 42, no. 4, pp. 463–484.

Haixiang, G. et al. (2017). "Learning from class-imbalanced data: Review of methods and applications". In: *Expert systems with applications* vol. 73, pp. 220–239.

Halsteinslid, E. L. (2019). *Addressing collinearity and class imbalance in logistic regression for statistical fraud detection*. eng.

Hanley, J. A. and McNeil, B. J. (1982). "The meaning and use of the area under a receiver operating characteristic (ROC) curve." In: *Radiology* vol. 143, no. 1, pp. 29–36.

Hastie, T. et al. (2009). *Elements of Statistical Learning: Data Mining, Inference, and Prediction*. eng. Springer series in statistics. New York: Springer.

— (2021). "An Introduction to glmnet". In: *CRAN R Repositary*.

He, H. and Edwardo A, G. (2009). "Learning from imbalanced data". In: *IEEE Transactions on knowledge and data engineering* vol. 21, no. 9, pp. 1263–1284.

Hoerl, A. E. and Kennard, R. W. (1970). "Ridge regression: Biased estimation for nonorthogonal problems". In: *Technometrics* vol. 12, no. 1, pp. 55–67.

Hosmer Jr, D. W. et al. (2013). *Applied logistic regression*. John Wiley & Sons, pp. 12, 177.

Japkowicz, N. and Stephen, S. (2002). "The class imbalance problem: A systematic study". In: *Intelligent data analysis* vol. 6, no. 5, pp. 429–449.

Kohavi, R. (1995). "A study of cross-validation and bootstrap for accuracy estimation and model selection". In: *Ijcai*. Vol. 14. 2. Montreal, Canada, pp. 1137–1145.

Krawczyk, B. (2016). "Learning from imbalanced data: open challenges and future directions". In: *Progress in Artificial Intelligence* vol. 5, no. 4, pp. 221–232.

Le Cessie, S. and Van Houwelingen, J. C. (1992). "Ridge estimators in logistic regression". In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* vol. 41, no. 1, pp. 191–201.

Marques, A. I., Garcia, V., and Sanchez, J. S. (2013). "On the suitability of resampling techniques for the class imbalance problem in credit scoring". eng. In: *The Journal of the Operational Research Society* vol. 64, no. 7, pp. 1060–1070.

Saito, T. and Rehmsmeier, M. (2015). "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets". In: *PloS one*, e0118432–e0118432.

Swets, J. A. (1988). "Measuring the accuracy of diagnostic systems". In: *Science* vol. 240, no. 4857, pp. 1285–1293.

Tibshirani, R. (1996). "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* vol. 58, no. 1, pp. 267–288.

— (1997). "The lasso method for variable selection in the Cox model". In: *Statistics in medicine* vol. 16, no. 4, pp. 385–395.

Tomek, I. (1976). "Two Modifications of CNN". eng. In: *IEEE transactions on systems, man, and cybernetics* vol. SMC-6, no. 11, pp. 769–772.

Van Hulse, J. et al. (2007). "Experimental perspectives on learning from imbalanced data". In: *Proceedings of the 24th international conference on Machine learning*, pp. 935–942.

Weiss, G. M. (2004). "Mining with rarity: a unifying framework". In: *ACM Sigkdd Explorations Newsletter* vol. 6, no. 1, pp. 7–19.

Weiss, G. M. and Provost, F. (2001). *The effect of class distribution on classifier learning: an empirical study*. Tech. rep. Rutgers University.

Zeng, X. and Martinez, T. R. (2000). "Distribution-balanced stratified cross-validation for accuracy estimation". In: *Journal of Experimental & Theoretical Artificial Intelligence* vol. 12, no. 1, pp. 1–12.

Zou, H. and Hastie, T. (2005). "Regularization and variable selection via the elastic net". In: *Journal of the royal statistical society: series B (statistical methodology)* vol. 67, no. 2, pp. 301–320.