Learning Imbalanced Data Sets with Noisy Replication

A Thesis

Presented in Partial Fulfilment of the Requirements for the

Degree of Master of Science

with a

Major in Statistical Science

in the

College of Graduate Studies

University of Idaho

by

Ensheng Dong

Major Professor: Stephen S. Lee, Ph.D.

Committee Member: Michelle M. Wiest, Ph.D.

Committee Member: Fuchang Gao, Ph.D.

Department Administrator: Christopher J. Williams, Ph.D.

May 2017

Authorization to Submit Thesis

This thesis of Ensheng Dong, submitted for the degree of Master of Science with a major in Statistical Science and titled "Learning Imbalanced Data Sets with Noisy Replication" has been reviewed in final form. Permission, as indicated by the signatures and dates given below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor ____________________________________ Date_______________

Stephen S. Lee, Ph.D.

Committee

Members ____________________________________ Date_______________

Michelle M. Wiest, Ph.D.

____________________________________ Date_______________

Fuchang Gao, Ph.D.

Department

Administrator ____________________________________ Date_______________

Christopher J. Williams, Ph.D.

Abstract

The noisy replication method has been proven to be an effective approach in learning the imbalanced binary data set in previous researches. This thesis expands its concept and effectiveness in broader scenarios: we study with several levels of sigma noise, a wide range of imbalanced ratios (IR), eight commonly used machine learning models, both binary and multi-class data sets, adding both noise and anti-noise, and more than 60 simulated and real data sets, etc. This thesis finds that the performance of the noisy replication method is significantly improved with the increase of IR by adding a relatively small noise for some models, KNN, Neural Network and C5.0, for instance. Moreover, it further shows that the noisy replication method is an ideal model-free approach in learning both the binary and the multi-class imbalanced data sets in terms of ROC area and Kullback-Leibler distance.

Acknowledgements

I would never have been able to finish my thesis without the guidance of my committee members, help from friends, and support from my family.

First and foremost, I would like to express my deepest and most sincere gratitude to my major advisor, Dr. Stephen Lee, who introduced me to the field of the machine learning, inspired me with this topic, and managed to help me with many difficulties in this thesis. I truly enjoyed Dr. Lee's class, STAT 504, which opened my eyes to the future of statistical learning. Because of this wonderful experience, I decided to do more researches in this field, and the noisy replication is undoubtedly a good beginning. There were many times that Dr. Lee and I discussed issues in his office on the 4th floor of Brink Hall, and every time I was enlightened with new methods. His support and encouragement are vital for my success.

I would also like to thank Dr. Michelle Wiest, Dr. Fuchang Gao, Dr. Christopher Williams and other professors from the Department of Statistical Science and the Department of Mathematics. It is in their classes that I learned how to solve a probability problem, to conduct a two-stage cluster sampling, to design an experiment with "Jalapeño Tofu hotdog", to analyze regression outcomes, to predict Ebola in West Africa with epidemiology study, to understand and practice the American Statistical Association's Ethical Guidelines, etc. It is in their classes that my understanding of statistics and data science significantly improved. It is in their classes that I received solid foundation on statistics and analysis for this thesis.

Special thanks go to Dr. Patrick Owsley, a former professor in engineering. Without Dr. Owsley's constant encouragement, I would never have completed this thesis. Many thanks to Dr. Felix Liao, a professor from the Department of Geography, who offered me tremendous help so that I could continue my master's study. Also many thanks to Chung Yan Wan, Brenda Henry, Dongyun Wang, Renae Shrum, Elizabeth

Ng, and many classmates and friends from the Department of Statistical Science and the Statistics Assistance Center. Their enthusiasm in statistics and optimistic view of life will be the great treasure for my whole life.

Finally, I would like to thank my family for always supporting me and encouraging me with their best wishes

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Imbalanced data, or skewed data, refers to a data set which has a dominant class much larger than other classes in number, or a data set which has one or more underrepresented classes [3, 4, 5]. During the past decade, there has been a significant improvement in the machine learning, data mining, and big data area. Lots of new methods and new applications are actively evolving at a fast pace. So is the study of imbalanced data set. Therefore, a good algorithm dealing with the imbalanced data set will have significant practical implications in many fields, such as finance, biology, medicine, telecommunication [6], and even terrorist detection. For instance, artificial intelligence scientists need to deal with the imbalanced data so as to recognize facial expressions [7]. The online advertising company may be interested in the click through rate in order to impress the audience [8]. In medical research, the number of patients of a rare disease is much fewer than common patients. To predict, prevent, and cure the disease need to face the imbalanced data issue. To solve this challenging machine learning problem [9], we introduce a method called *noisy replication*.

In this thesis, we refer to the dominate class as the *majority class*, and the class with the least number of observations as the *minority class*. Typically, for an imbalanced data set, the number of observations in the minority class are far smaller than one or more other classes. For an imbalanced binary data set, it can be expressed as $T = \{(x_i, y_i), i = 1, ..., n_0 + n_1\} = \{(x_i, 0), i = 1, ..., n_0\} \cup \{(x_i, 1), i = 1, ..., n_1\}$, where $n_0 >> n_1$. The class whose response variable equals to 0 is the majority class, while the class with the response variable equals to 1 is the minority class. For a data set containing three or more classes, the majority class is the class with the largest observations, while the minority class has the least number of observations. The imbalance ratio (IR) is defined as the the number of the majority class over the number of the minority class. The KEEL (Knowledge Extraction based on Evolutionary Learning) data set classifies with two values IR = 1.5 and IR = 9.0 [10]. Therefore, we adopt a similar principle, and any data set with IR is equal to or greater than 1.5 will be defined as an imbalanced data set, regardless of the number of total classes.

A variety of researches have been done in learning the imbalanced data. He and Garcia (2009) summarized many methods for learning the imbalanced data set, such as the oversampling/undersampling method, the cost-sensitive method, the kernel-based method, etc. [5] This thesis mainly concentrates on proving the effectiveness of a machine learning algorithm, the noisy replication method, in predicting and classifying the imbalanced data set with two or more classes. The basic principle of the noisy replication method is to add a slight noise to the minority class, and to duplicate the minority observations several times, so as to lower the skewness of the data set and increase the success rate of prediction and classification. That means after applying this method, the data set should be $T = \{(x_i, y_i), i = 1, ..., n_0 + n_1\} = \{(x_i, 0), i = 1, ..., n_0\} \cup \{(x_i + \epsilon_{ij}, 1), i = 1, ..., n_1 \times m\}$, where ϵ_{ij} refers to the noise added to each observation in the minority class, and it should be scaled according to the value of that observation. If the observation value is large, then ϵ_{ij} should be increased, and vice versa. In addition, m refers to the number of duplications. Consequently, the total number of observations is $(n_0 + n_1 \times m)$ after applying the noisy replication method. That means we increase the number of observations in a reasonable way, decrease the skew of the data set, and increase the success rate to predict the imbalanced data set.

The noisy replication method has been proved to be effective for imbalanced binary data sets. Lee (1999) first demonstrated that by adding noisy replicates to the minority class, the prediction performance of several classification models could be improved [11]. Lee (2000) then improved the algorithm by increasing replications of the minority part, and adding noise to the training data set [4]. To further expand this methodology, we tested both binary and multi-class data set with different levels of noise in eight commonly used machine learning models: K-Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), Logistic Regression (Logistic), Support Vector Machine (SVM), Neural Network (Neural), Naïve Bayes (NB), C5.0, and Partial Least Squares Discriminant Analysis (PLS). The cross-validation method is also applied for a more accurate result. This thesis will continue the research on noisy replication methodology, and bring its application to a broader scope. At the end of this thesis, we will get clearer answers to the following questions:

1. What kind of noise should be added?
2. Where should the noise be added, training data set, testing data set or both?
3. How many times should the minority classes be repeated?
4. Will anti-noise, or two-way vibration improve the performance?
5. Can this algorithm be applied to both the qualitative and the quantitative data?
6. Will this method be applied to the data set with multiple classes?
7. Will the imbalance ratio (IR) influence on the model performance?
8. How to measure and assess the performance of the algorithm, such as ROC area and Kullback-Leibler distance? Which one is better?
9. Which model performs better with the noisy replicates?
10. How good is this method compared with other algorithms using the same real data set?

There are six chapters in this thesis. Followed by this introduction chapter, Chapter 2 introduces eight commonly used machine learning models, and the performance metrics of the algorithm. Chapter 3 further explains the noisy replication method with the pseudocode, and tests this method with a simulated highly imbalanced binary data set. Chapter 4 continues the testing with about fifty (50) real binary data sets getting from the KEEL website. Chapter 5 expands the application of noisy replication method to multi-class imbalanced data set. Both a simulated data set and ten (10) real data sets are tested. Chapter 6 concludes our findings and points out the direction for future researches.

Chapter 2

Model Selection

Two types of variables are often studied: quantitative (also called continuous or numerical) data and qualitative (categorical) data. The quantitative variable measures the quantity of an observation, such as age, weight, height, income, etc. The qualitative variable approximates or characterizes the attributes of an observation into different categories, such as gender, education level, ethnicity, diagnosis result (positive or negative), etc. Both types of variables are widely used in scientific research as well as in business survey. This thesis studies the qualitative data. Hence, a data set could have either two classes or more than two classes (multi-class). We are interested in predicting to which class the new observation belongs, regardless of the number of classes in the dependent variable. Both classification methods and regression methods will be adopted.

This chapter lays the theoretical foundation in order to address the goal of this thesis. We first introduced eight commonly used machine learning models: 10-Nearest Neighbors (10NN), Linear Discriminant Analysis (LDA), Logistic Regression (Logistic), Support Vector Machine (SVM), Neural Network (Neural), Naïve Bayes (NB), C5.0, and Partial Least Squares Discriminant Analysis (PLS). After that, we described two criteria in measuring the performance of these models: the Receiver Operating Characteristic (ROC) method and the Kullback–Leibler Distance (KL distance) method. An indispensable tool in model statistics, Cross-Validation (CV), is also discussed in the end of this chapter.

2.1 K-Nearest Neighbors (KNN)

Given a testing observation x_0, the closest K training observations near x_0 are selected, and the classification of x_0 is defined by the largest probability of these K training observation. This is called K-nearest neighbors (KNN) method. A lower K value corresponds to a data set which has low bias and very high variance. A higher K value corresponds to a data set which has low variance but high bias. When K = 0, the training error rate is 0, but the testing error rate should be higher. Therefore, while KNN is one of the simplest classification methods, it can also make highly accurate predictions if each class of the data set has very dissimilar feature values. We will adopt the `kknn` function from the R package "kknn" [12] with K = 10.

2.2 Logistic Regression

The general linear model (GLM) could be an easy and straightforward solution of predicting the quantitative data; however, when the response variable becomes the qualitative data, a better classification method (classifier) should be deployed, and logistic regression is one of the solutions. Similar to KNN, logistic regression is a widely-used classifier. If the response variable is binary 0 or 1, the probability of the response variable will not be below 0 or above 1, while the general linear regression may generate a probability prediction below 0 or above 1, which could be contradictory to the reality. The logistic function looks like a general linear regression function:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

where X is the observation data set. p is the probability of the observed data sets, ranging between 0 and 1. The form of the logistic function is S-shaped. Intercept and slope could be calculated by maximum likelihood method. For a two-class or binary qualitative response dataset with multiple predictors, the multiple logistic regression function could be built (or multinomial logistic regression):

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

where X_i represents different training data sets. The logistic regression could also be extended to data sets that contain multiple response classes. Compared with the SVM, mentioned below, the logistic regression is more simplistic in estimating the classification boundary [8]. The most frequently used functions in R are `multinom` (from the package nnet) [13] and `mlogit` (from the package mlogit) [14].

2.3 Linear Discriminant Analysis (LDA)

When there are more than two (2) response classes, the linear discriminant analysis (LDA) could serve as a more stable and popular method than the logistic regression [1]. The linear discriminant analysis is like the principal component analysis (PCA), but it focuses on maximizing the separability among all known classes. In this thesis, the `lda` function in the R package "MASS" will be applied [15].

2.4 Support Vector Machines (SVM)

The support vector machine (SVM) is another supervised classification approach. The main idea of SVM is to find a *hyperplane*, a flat subspace, to separate the training data set into two classes. The *maximal margin hyperplane* has the farthest perpendicular distance to each side of the training observations (Figure 2.1). However, there are some circumstances where the training data set is inseparable. To solve this problem, the *support vector classifier* makes some "violations" by allowing some observations to fall into the incorrect side of the margin or even the hyperplane. To adjust the support vector classifier, several tuning parameters are introduced. For instance, C determines the severity of the violation; polynomial kernel of degree d and radial kernel γ adjust the performance of the SVM, an extension of support vector classifier accommodating a non-linear boundary between two classes.

As for multi-class data set, there are two methods: *one-versus-one* classification

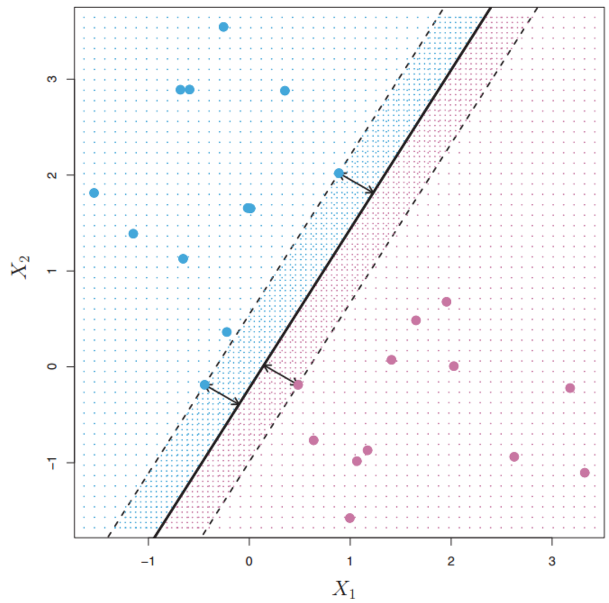


Figure 2.1: Maximal margin hyperplane [1]

and *one-versus-all* classification. To simplify this approach, this thesis adopts the `svm` function in the R package "e1071" [16] in dealing with both binary and multi-class data sets.

2.5 Neural Network

The basic idea of the neural network algorithm is to simulate human brain neuro nodes and connections inside a computer, so as to make the computer to learn data, and even to make decisions in a way like human beings. In this thesis, the `nnet` function in the R package "nnet" will be applied [17]. By default, the number of units in the hidden layer (`size`) is set as 2, and the maximum number of iterations (`maxit`) is set as 200.

2.6 Naïve Bayes

Naïve Bayes is another well-known classification method. Given the class veriable y and the dependent feature vector x_i, the Naïve Bayes rule could be expressed as

$$P(y|x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, \ldots, x_n)}$$

To compute the posterior probabilities of a discrete data set, we will use the function `naiveBayes` in an R package called "e1071" [16].

2.7 C5.0

C5.0 algorithm is a widely used decision tree method in machine learning. Initially people have ID3.0 algorithm. Based on ID3.0, C4.5 algorithm and C5.0 algorithm were developed finally [8]. This type of decision tree model is based on entropy and information gain. *Entropy* measures the impurity, and it controls where to split the data. If $Entropy = 0$, all examples are the same class. If $Entropy = 1$, examples are evenly split between classes. *Information gain* is based on entropy. The higher the value of information gain, the better C5.0 performs. In this thesis, the `C5.0` function in the R package "C50" will be applied [18].

2.8 Partial Least Squares Discriminant Analysis (PLS-DA)

Partial least squares (PLS) is an algorithm which could be used to predict both quantitative and qualitative variables. Classification with PLS is termed PLS-DA, where the DA stands for discriminant analysis. PLS is also a dimension reduction method. In this thesis, the `plsda` function in the R package "caret" will be applied [19].

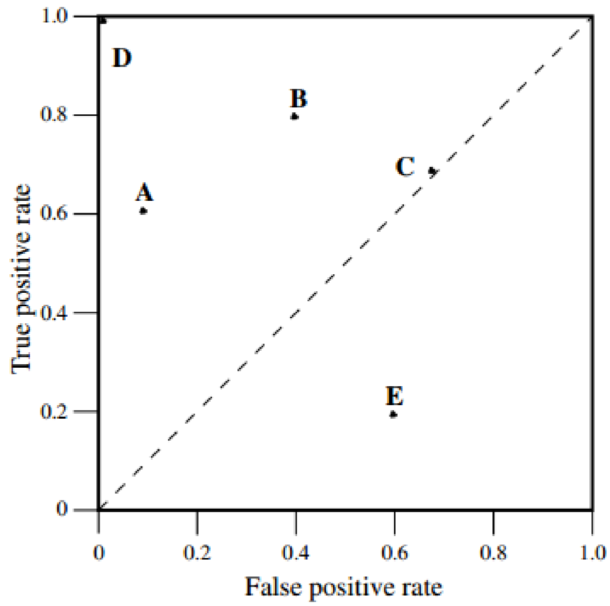


Figure 2.2: A basic ROC graph [2]

2.9 Assessment Criteria

2.9.1 Receiver Operating Characteristic (ROC)

The ROC (Receiver Operating Characteristics) curve is an ideal tool for visualizing and measuring the machine learning model results regardless of the class skew of the data set [2, 7]. For a binary data set, there are four possible classification outcomes. If the true class is positive and the prediction is positive, we call it *true positive (TP)*; if the true class is positive but the prediction is negative, we call it *false negative (FN)*; if the true class is negative and the prediction class is also negative, we call it *true negative (TN)*; if the true class is negative but the prediction class is positive, we call that *false positive (FP)*. The *true positive rate (benefits)* is defined as the number of positives correctly classified over the number of total positive, and the *false positive rate (costs)* is defined as the number of positives incorrectly classified over the number of total negative.

The ROC curve uses the false positive rate (0 - 1) as the x-axis, and the true positive rate (0 - 1) as the y-axis. If a prediction has a higher true positive rate and a lower false positive rate, its positive will be closer to (0, 1). Figure 2.2 is a brief graph of the ROC curve. Point D is the point with the best classification results. Point A is more conservative (less liberal) than B, since Point A has a relatively higher true positive rate and a relatively lower false positive rate than Point B. Points laying on the diagonal line, such as Point C, have a virtually random performance, or the prediction is made by guessing. Point E performs even worse than random guessing, which means the prediction is less useful.

The overall performance of an ROC curve could be measured by AUC (area under the ROC curve). The closer the ROC curve to the Point D, the larger the AUC value, and the better the classification model.

If there are more than two classes (multi-class) in the data set, it is hard to draw an ROC curve, however the AUC is still measurable with some techniques, such as the pairwise discrimination [20]. We could generate an AUC value for each pair of classes, and average the multiple AUC values as the multi-class AUC value. This technique is adopted in Chapter 5.

2.9.2 Kullback–Leibler Distance (KL)

The Kullback-Leibler distance, or KL divergence or KL distance, measures the "discrepency" or the "distance" between two models [21]. For the discrete distribution, the KL distance is defined as

$$D_{KL} = \sum p(\mathbf{x_i}) \log \frac{p(\mathbf{x_i})}{q(\mathbf{x_i})}$$

where $\mathbf{x_i}$ are observations with class y_i, $p(\mathbf{x_i})$ and $q(\mathbf{x_i})$ are discrete probability distributions. Both $p(\mathbf{x_i})$ and $q(\mathbf{x_i})$ sum up to 1. The smaller the KL distance is, the closer two models are.

2.10 Cross-Validation (CV)

Cross-validation is a resampling method in the statistics learning. K-fold cross-validation means randomly dividing observations into k groups, set one group as the validation data set, and the remaining $k-1$ groups as the training data set, so as to compute a more accurate assessment value, such as ROC and KL distance in our case. Specifically, if $k=n$, the number of observations, we call it *Leave-one-out cross-validation* (LOOCV). The reason to adopt the cross-validation method is to minimize the distinction between the *test error rate* and the *training error rate*, since a machine learning method may generate a relatively low training error rate, but it may also generate a relatively high testing error rate.

Considering the minority class of some data sets could contain less than 10 observations, we will adopt 2-fold cross-validation for testing both binary and multi-class data sets. We will also apply `nsim` = 100 times of cross-validations for each data set.

Chapter 3

Noisy Replication for Imbalanced Binary Data Sets

The fundamental idea of the noisy replication method is to add a small amount of noise to the minority class so that to improve the correct rate of the prediction. To testify its effectiveness, we plan to have a pilot experiment with a simulated imbalanced binary data set in this chapter. Details of the noisy replication with its pseudocode are described, and outcomes are analyzed.

3.1 Binary Data Set Simulation

The simulation is a good method to preview the possible outcome intuitively. Hence, we first simulate a binary data set with the imbalance ratio (IR) at 10.0. In the machine learning, training data set refers to the known observations, and it is used to teach a model (classifier) to predict the relationship between the independent and dependent variables, or between the respondent and explanatory variables. Our simulated training data set is defined as:

$$df.train = \left\{ (x_i, 0), x_i \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right), i = 1, ...200)) \right\} \cup$$

$$\left\{ (x_i, 1), x_i \sim N\left(\begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}, \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \right), i = 1, ...20)) \right\}$$

where the size of the data frame is 220, among which 200 observations are in the majority class and only 20 are in the minority class. Sometimes the majority class is called as Class 0, and the minority class as Class 1. We also increase the skew of the training data set by defining the mean and the covariance of the explanatory vectors of Class 0 and Class 1 very close to each other.

The testing data set, or the validation data set, is used to evaluate the performance of the trained model. Our testing data set is defined as:

$$df.test = \left\{ (x_i, 0), x_i \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right), i = 1, ...200)) \right\} \cup$$

$$\left\{ (x_i, 1), x_i \sim N\left(\begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}, \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \right), i = 1, ...20)) \right\}$$

where the size of the data frame, the majority class, and the minority class are 220, 200, and 20 respectively, the same as the training data set. The imbalance ratio (IR) could reflect the skew of the data set, and it is defined as the number of observations in the majority class over that of the minority class. Though the IR is a non-negative value, to generate a simulation prototype, both the training data set and the testing data set define the IR equals to 10.0.

3.2 Simulation Algorithm

Figure 3.1 illustrates the main idea of the noisy replication method: vibrate the duplicated minority class with noises in the training data set. In Figure 3.1, `noisy.repl` refers to the number of replications of the minority classes, and `noisy.train` refers to the number of training data sets after adding the noise, which is also called the *noisy training data set*. The noise is defined as $\varepsilon \sim N_q(0, \sigma^2_{noise}\Sigma_q)$, where Σ_q is the diagonal variance matrix of the duplicated minority classes, and q is the size of duplicated minority class. For instance, when `noisy.repl` $= 1$, and the size of the minority class is 20, $q =$ `noisy.repl` $\times$ 20. The noise level, σ_{noise} (`sigma.noise`), is selected from 0.1, 0.5, and 1.0. One of the objectives of this simulation is to compare among three noise levels to evaluate which `sigma.noise` performs better.

In each experiment, we first simulate a training data set (the light yellow part)

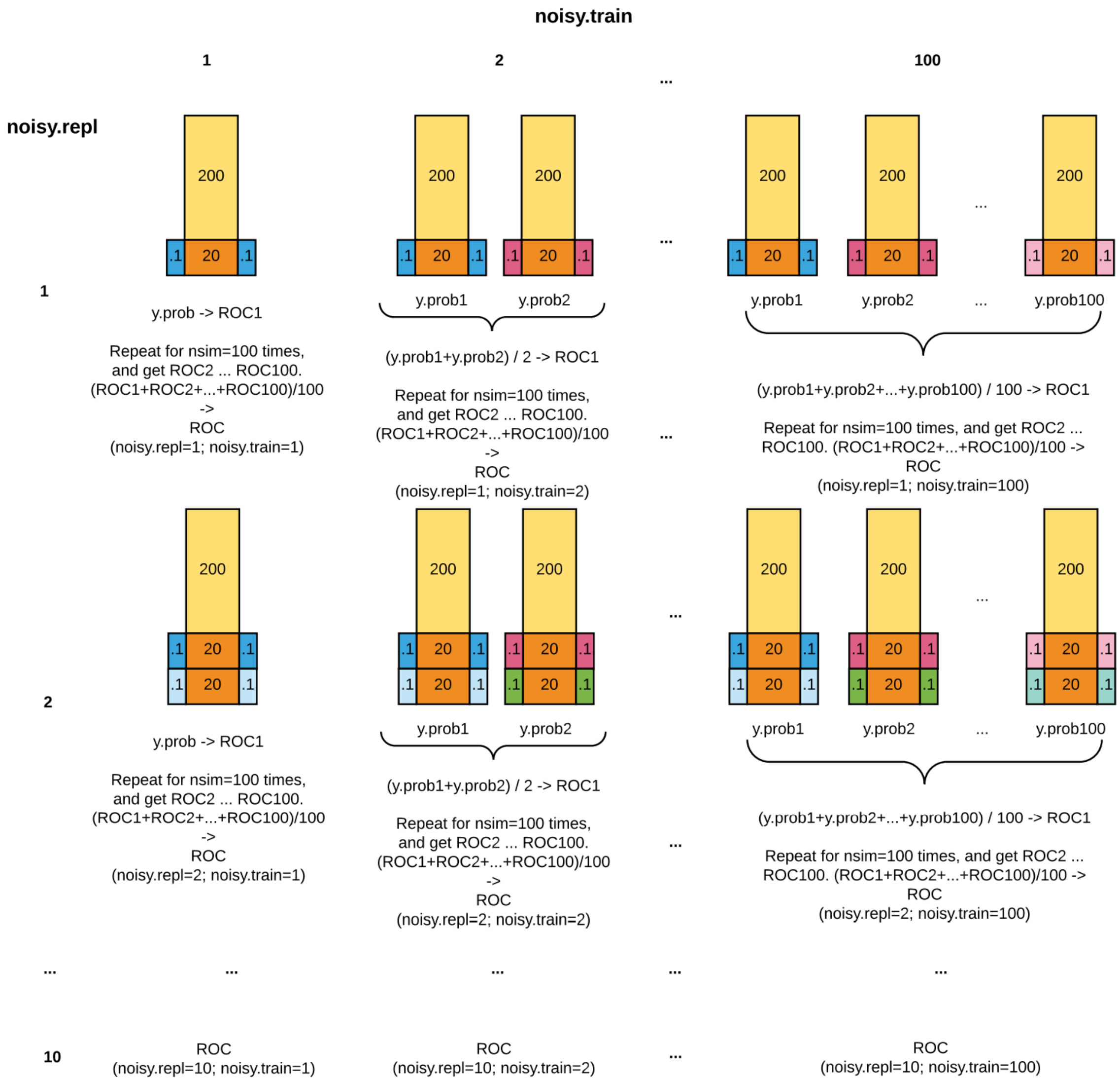


Figure 3.1: The noisy replication method explained, using ROC and `sigma.noise` = 0.1 as an example

and a testing data set (the orange part). Values for ROC/AUC and KL distance are calculated as the original assessment criteria, expressed as `ROC.original` and `KL.original`. When adding the noise to the training data set for the first time, we can receive another group of values for ROC/AUC and KL distance. Repeating for `nsim` = 100 times with different noises, we finally receive 100 ROC/AUC values and 100 KL distance values. Averaging these 100 values, and we get improved values for ROC/AUC and KL distance with the noisy replication method. In Figure 3.1, `ROC(noisy.repl=1; noisy.train=1)` represents this improved ROC/AUC value after the applying the noise replication method. We also use $\triangle$ROC and $\triangle$KL to express the difference in ROC value and KL distance before and after applying the noisy replication method respectively.

When `noisy.repl` $= 1$ and `noisy.train` $= 2$, we duplicate the training data set twice, and apply the noisy replication method to each individual training data set. Repeating for `nsim` $= 100$ times with different noises, we could also receive the improved ROC/AUC values and 100 KL distance values. In Figure 3.1, `ROC(noisy.repl=1; noisy.train=2)` represents another improved ROC/AUC value after the applying the noise replication method. The maximum `noisy.train` is set as 100 in this simulation experiment.

When `noisy.repl` $= 2$ and `noisy.train` $= 1$, we duplicate the minority class twice. Therefore, the size of the training data set becomes 240. The noise will then be added to the minority class, whose size is 40. Repeating for `nsim` $= 100$ times with different noises, we could also receive the improved ROC/AUC values and 100 KL distance values. In Figure 3.1, `ROC(noisy.repl=2; noisy.train=1)` represents another improved ROC/AUC value after the applying the noise replication method. The maximum `noisy.repl` is set as 10 in this simulation experiment.

There are two types of vibration: one-side and two-side. For the one-side vibration, a noise is simply added to the minority class. For the two-side vibration, the same noise will be first added to the minority class and calculate the posterior probability. Then the same noise will be deleted from the same minority class, whose posterior probability will be calculated separately. The classifier assessment is based on the mean of these two posterior probabilities. In Figure 3.1, the noise is represented with a smaller box adjacent to the minority orange box. The number in the noise box means the noise level. In this simulation experiment, both the one-side vibration and two-side vibration, and three levels of noise are tested.

3.3 Pseudocode

A pseudocode explaining the noisy replication method is presented in the next page, using the simulated binary data set with IR=10.0 as an example.

```
Data: models = {KNN, LD, Logistic, SVM, Neural, NB, C5.0, PLS}
      sigma.noise = {0.1, 0.5, 1.0}
      noisy.repl = number of minority classes replications {1 to 10}
      noisy.train = number of training data sets {10, 20, 40, 60, 80,
      100}
      nsim = number of simulation times, defined as 100
Result: ROC/AUC and KL tables with noisy.repl = 10, noisy.train = 100,
        sigma.noise = {0.1, 0.5, 1.0}, nsim = 100 times, and 10-fold CV.

for each model do
    for each sigma.noise do
        for nsim = 100 times do
            for each k-fold cross-validation do
                Generate ROC and KL for the original data set;
                for each noisy.repl do
                    Duplicate the minor class for the training data set;
                    for each noisy.train do
                        Add noise to each minor class in the training data set
                          and get a new training data set;
                        1) One-side vibration noise;
                        2) Two-side vibration noise;
                        for each model do
                            1) Calculate prediction probabilities for each
                              vibrated training data set with one-side vibration;
                            2) Calculate prediction probabilities for each
                              vibrated training data set with two-side vibration;
                            Calculate the vibrated ROC and KL based on the
                              average prediction probabilities;
                            Calculate the difference between the original and
                              the vibrated assessments;
                        end
                        Record the assessment difference for all replications of
                          the training data set;
                    end
                    Record the assessment difference for all replications of the
                      minor classes;
                end
                Average the difference table over k.cv = 10;
            end
            Average the difference table by nsim = 100;
        end
        Go to the next sigma.noise;
    end
    Go to the next model;
    Generate the output assessment tables and their plots;
end
```

3.4 Simulation Results and Interpretations

The following tables and figures summarize outcomes of this pilot experiment. We categorize them into four groups.

- Group 1: Table 3.1 and Figure 3.2 outline the results when `noisy.repl` = 1, `noisy.train` = 100, `nsim` = 100, and vibration = one-side.
- Group 2: Table 3.2 and Figure 3.3 outline the results when `noisy.repl` = 1, `noisy.train` = 100, `nsim` = 100, and vibration = two-side.
- Group 3: Table 3.3 and Figure 3.4 outline the results when `noisy.repl` = 3, `noisy.train` = 100, `nsim` = 100, and vibration = one-side.
- Group 4: Table 3.4 and Figure 3.5 outline the results when `noisy.repl` = 3, `noisy.train` = 100, `nsim` = 100, and vibration = two-side.

Each table has the results for eight machine learning models with three noise levels (σ_{noise}) respectively. **ROC0** refers to the original ROC/AUC value before applying the noisy replication method; **ROC100** refers to the ROC/AUC value after adding noise replicates with `noisy.train` = 100; **ROC.diff** is $\triangle$ROC; **KL0** refers to the original KL distance value before applying the noisy replication method; **KL100** refers to the ROC/AUC value after adding noise replicates with `noisy.train` = 100; **KL.diff** is $\triangle$KL distance.

Each figure has two subgraphs, and each subgraph has eight plots for 95% confidence intervals of $\triangle$ROC and $\triangle$KL distance after applying the noisy replication method. The x-axis is three noise levels, and the y-axis is $\triangle$ROC (`ROC.diff`) or $\triangle$KL distance (`KL.diff`). If there is no significant difference between the original model and the basic model and the noisy replication model, then 95% confidence interval will contain $\triangle$ROC = 0 or $\triangle$KL distance = 0. If the entire interval is positive (i.e., above $\triangle$ROC = 0 or below $\triangle$KL distance), then we can say that the noisy replication makes a statistically significant difference compared with the regular models. The means of $\triangle$ROC or $\triangle$KL distance are joined by the solid line.

Group 1 demonstrates the following models performing better after adding noise replicates for most noise levels: KNN, Logistic Regression, Neural Network, and C5.0 (for $\triangle$ROC); KNN, Linear Discriminant Analysis, Logistic Regression, SVM, and Neural Network (for $\triangle$KL distance).

Group 2 demonstrates the following models performing better after adding noise replicates for most noise levels: KNN, Neural Network, and C5.0 (for $\triangle$ROC); KNN, Linear Discriminant Analysis, Logistic Regression, SVM, and Neural Network (for $\triangle$KL distance).

Group 3 demonstrates the following models performing better after adding noise replicates for most noise levels: SVM, and C5.0 (for $\triangle$ROC); KNN, and Neural Network (for $\triangle$KL distance).

Group 4 demonstrates the following models performing better after adding noise replicates for most noise levels: SVM, Neural Network and C5.0 (for $\triangle$ROC); KNN, Linear Discriminant Analysis, Logistic Regression, KNN, and Neural Network (for $\triangle$KL distance).

Comparing Group 1 and 3 or Group 2 and 4, we can examine which vibration method is better: one-side or two-side. Comparing Group 1 and 2 or Group 3 and 4, we can examine how many `noisy.repl` to select for the following experiment with real data sets. To conclude, both one-side and two-side vibration have similar outcomes, and `noisy.repl` = 1 is better than `noisy.repl` =3. Hence, we decide to use the following parameters to test all other simulated and real data sets in this thesis: `nsim` = 100, `noisy.repl` = 1, `noisy.train` = 100, `sigma.noise` = (0.1, 0.5, 1.0), and vibration = two-side.

Table 3.1: Pilot simulation summary with one-side vibration and `noisy.repl` = 1

Model	σ_{noise}	ROC0	ROC100	ROC.diff	KL0	KL100	KL.diff
KNN	0.1	0.73	0.79	0.05	10.48	4.22	-6.26
	0.5	0.75	0.80	0.05	10.42	2.87	-7.55
	1.0	0.72	0.76	0.04	10.48	2.82	-7.65
LDA	0.1	0.81	0.81	0.00	0.24	0.24	0.00
	0.5	0.81	0.81	0.00	0.24	0.24	0.00
	1.0	0.81	0.81	0.00	0.24	0.24	0.00
Logistic	0.1	0.81	0.81	0.00	0.24	0.24	0.00
	0.5	0.81	0.81	0.00	0.24	0.24	0.00
	1.0	0.81	0.81	0.00	0.24	0.24	0.00
SVM	0.1	0.64	0.64	0.00	0.28	0.27	-0.01
	0.5	0.66	0.63	-0.03	0.27	0.27	0.00
	1.0	0.65	0.62	-0.03	0.27	0.27	0.00
Neural	0.1	0.80	0.83	0.03	0.28	0.24	-0.04
	0.5	0.80	0.82	0.02	0.29	0.24	-0.05
	1.0	0.79	0.81	0.03	0.31	0.24	-0.07
NB	0.1	0.81	0.80	0.00	0.25	0.25	0.00
	0.5	0.81	0.80	-0.01	0.25	0.24	0.00
	1.0	0.82	0.79	-0.03	0.24	0.25	0.01
C5.0	0.1	0.62	0.73	0.11	0.29	0.46	0.17
	0.5	0.63	0.73	0.10	0.29	0.40	0.11
	1.0	0.60	0.71	0.11	0.29	0.37	0.08
PLS-DA	0.1	0.80	0.80	0.00	0.42	0.42	0.00
	0.5	0.81	0.81	0.00	0.42	0.42	0.00
	1.0	0.81	0.81	0.00	0.42	0.42	0.00

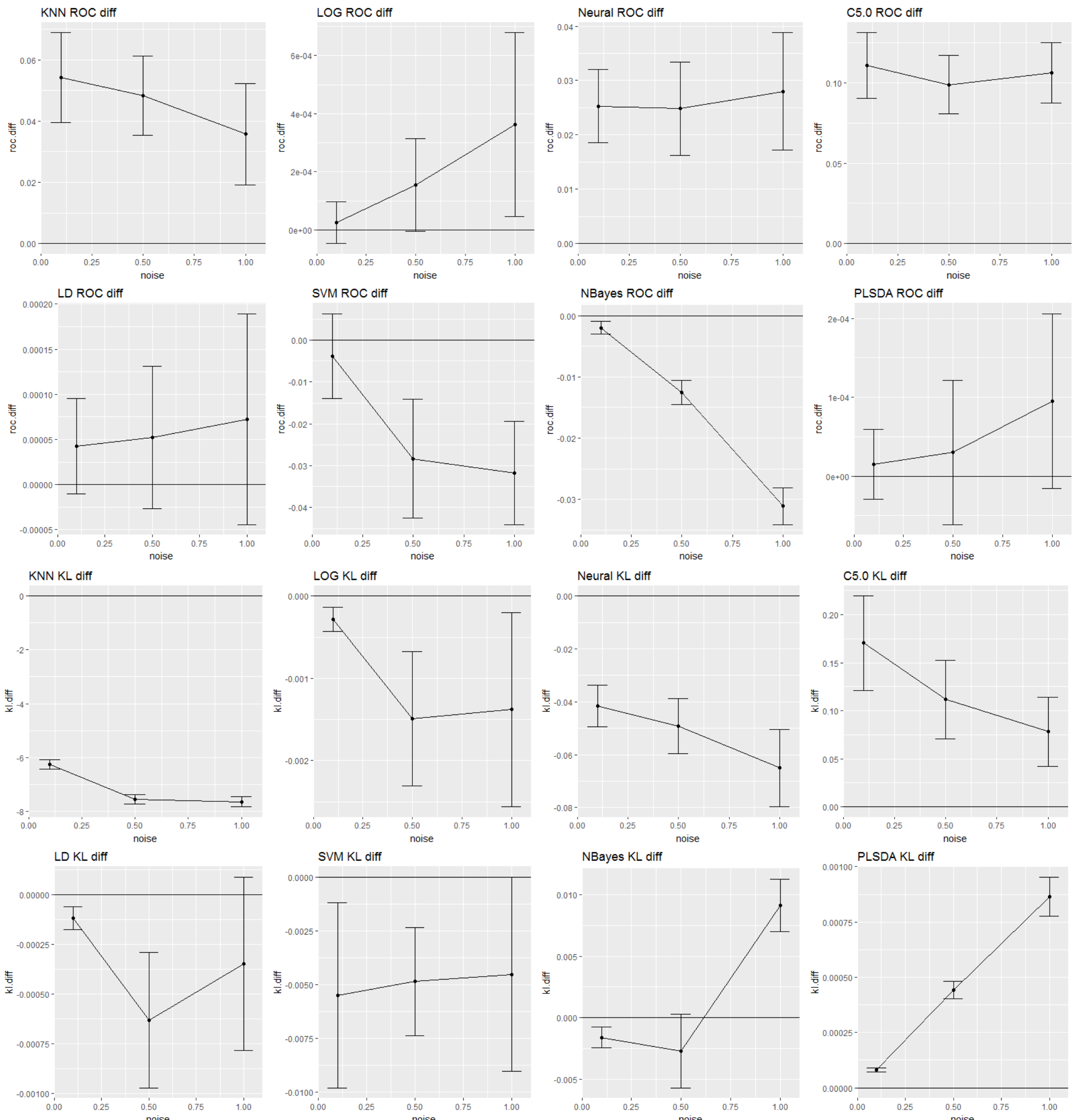


Figure 3.2: Pilot simulation outcome with one-side vibration and `noisy.repl` = 1

Table 3.2: Pilot simulation summary with two-side vibration and `noisy.repl` = 1

Model	σ_{noise}	ROC0	ROC100	ROC.diff	KL0	KL100	KL.diff
KNN	0.1	0.74	0.79	0.05	10.59	4.30	-6.29
	0.5	0.73	0.79	0.06	10.37	2.86	-7.51
	1.0	0.76	0.78	0.02	10.48	2.81	-7.67
LDA	0.1	0.80	0.80	0.00	0.24	0.24	0.00
	0.5	0.82	0.82	0.00	0.24	0.24	0.00
	1.0	0.82	0.82	0.00	0.24	0.24	0.00
Logistic	0.1	0.82	0.82	0.00	0.24	0.24	0.00
	0.5	0.80	0.80	0.00	0.25	0.24	0.00
	1.0	0.80	0.80	0.00	0.24	0.24	0.00
SVM	0.1	0.65	0.64	-0.01	0.27	0.27	0.00
	0.5	0.65	0.63	-0.03	0.28	0.27	-0.01
	1.0	0.66	0.62	-0.04	0.28	0.27	-0.01
Neural	0.1	0.79	0.82	0.03	0.30	0.24	-0.06
	0.5	0.77	0.81	0.04	0.31	0.25	-0.06
	1.0	0.79	0.81	0.03	0.30	0.24	-0.06
NB	0.1	0.81	0.81	0.00	0.24	0.24	0.00
	0.5	0.81	0.79	-0.01	0.25	0.24	0.00
	1.0	0.81	0.78	-0.03	0.25	0.25	0.00
C5.0	0.1	0.60	0.73	0.12	0.29	0.45	0.16
	0.5	0.62	0.73	0.11	0.29	0.34	0.05
	1.0	0.60	0.72	0.12	0.29	0.36	0.07
PLS-DA	0.1	0.81	0.81	0.00	0.42	0.42	0.00
	0.5	0.81	0.81	0.00	0.42	0.42	0.00
	1.0	0.81	0.81	0.00	0.42	0.42	0.00

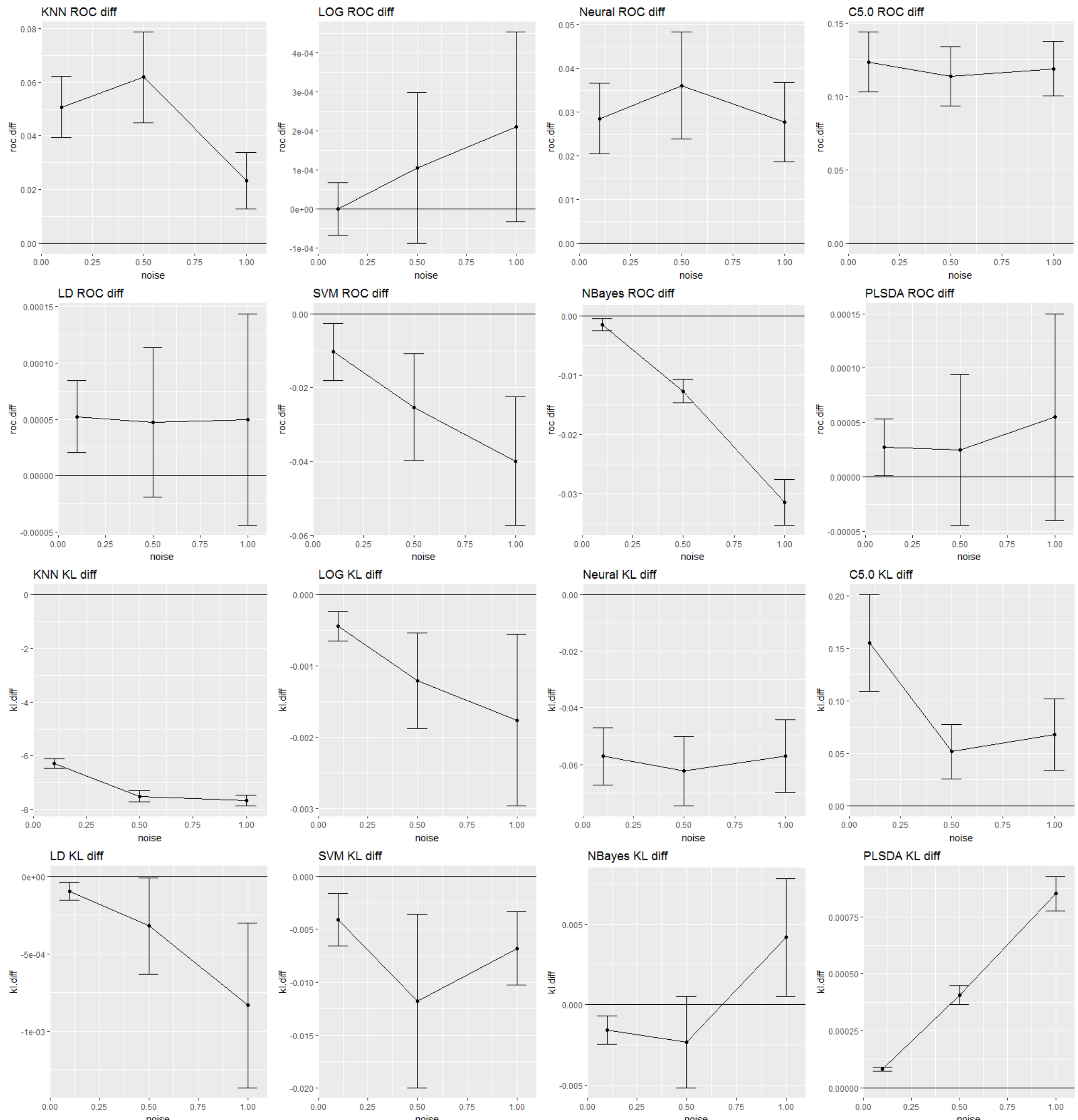


Figure 3.3: Pilot simulation outcome with two-side vibration and `noisy.repl` = 1

Table 3.3: Pilot simulation summary with one-side vibration and `noisy.repl` = 3

Model	σ_{noise}	**ROC0**	**ROC100**	**ROC.diff**	**KL0**	**KL100**	**KL.diff**
KNN	0.1	0.73	0.69	-0.04	10.39	3.01	0.10
	0.5	0.72	0.70	-0.02	10.43	1.98	0.50
	1.0	0.73	0.70	-0.03	10.59	1.95	1.00
LDA	0.1	0.82	0.82	0.00	0.24	0.29	0.10
	0.5	0.81	0.81	0.00	0.24	0.29	0.50
	1.0	0.82	0.82	0.00	0.24	0.28	1.00
Logistic	0.1	0.81	0.81	0.00	0.24	0.29	0.10
	0.5	0.83	0.83	0.00	0.23	0.29	0.50
	1.0	0.81	0.81	0.00	0.24	0.29	1.00
SVM	0.1	0.66	0.75	0.09	0.27	0.30	0.10
	0.5	0.65	0.72	0.07	0.28	0.31	0.50
	1.0	0.67	0.70	0.04	0.27	0.30	1.00
Neural	0.1	0.78	0.81	0.03	0.30	0.29	0.10
	0.5	0.78	0.80	0.02	0.31	0.29	0.50
	1.0	0.79	0.79	0.00	0.31	0.29	1.00
NB	0.1	0.81	0.81	0.00	0.25	0.29	0.10
	0.5	0.80	0.79	-0.01	0.25	0.29	0.50
	1.0	0.81	0.78	-0.03	0.25	0.29	1.00
C5.0	0.1	0.62	0.76	0.14	0.29	0.70	0.10
	0.5	0.61	0.75	0.14	0.29	0.66	0.50
	1.0	0.62	0.73	0.11	0.29	0.74	1.00
PLS-DA	0.1	0.81	0.81	0.00	0.42	0.47	0.10
	0.5	0.82	0.82	0.00	0.42	0.47	0.50
	1.0	0.82	0.82	0.00	0.42	0.47	1.00

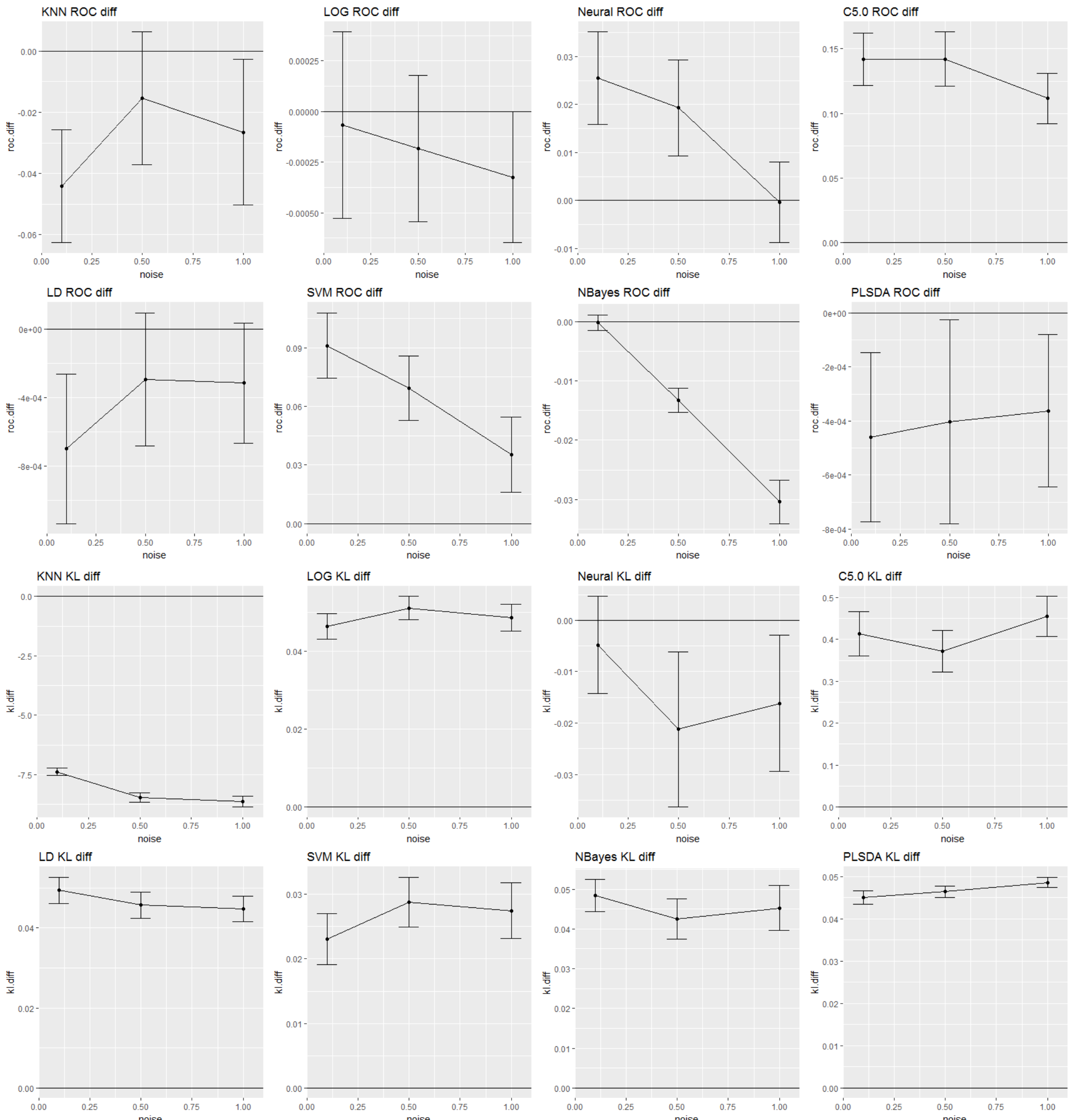


Figure 3.4: Pilot simulation outcome with one-side vibration and `noisy.repl` = 3

Table 3.4: Pilot simulation summary with two-side vibration and `noisy.repl` = 3

Model	σ_{noise}	ROC0	ROC100	ROC.diff	KL0	KL100	KL.diff
KNN	0.1	0.74	0.67	-0.07	10.59	3.13	-7.47
	0.5	0.74	0.70	-0.04	10.46	1.97	-8.48
	1.0	0.74	0.69	-0.05	10.49	1.95	-8.54
LDA	0.1	0.82	0.81	0.00	0.24	0.29	0.05
	0.5	0.81	0.81	0.00	0.24	0.29	0.04
	1.0	0.81	0.81	0.00	0.24	0.28	0.04
Logistic	0.1	0.81	0.81	0.00	0.24	0.29	0.05
	0.5	0.82	0.82	0.00	0.24	0.29	0.05
	1.0	0.80	0.80	0.00	0.25	0.30	0.05
SVM	0.1	0.66	0.75	0.09	0.28	0.30	0.02
	0.5	0.65	0.73	0.07	0.27	0.30	0.03
	1.0	0.66	0.70	0.05	0.27	0.30	0.03
Neural	0.1	0.79	0.81	0.03	0.29	0.29	0.00
	0.5	0.78	0.81	0.03	0.31	0.29	-0.02
	1.0	0.78	0.80	0.01	0.30	0.29	-0.01
NB	0.1	0.82	0.82	0.00	0.24	0.29	0.05
	0.5	0.81	0.80	-0.01	0.25	0.29	0.04
	1.0	0.81	0.78	-0.03	0.24	0.29	0.05
C5.0	0.1	0.61	0.77	0.16	0.29	0.67	0.38
	0.5	0.61	0.76	0.15	0.29	0.59	0.30
	1.0	0.61	0.74	0.12	0.29	0.68	0.39
PLS-DA	0.1	0.82	0.82	0.00	0.42	0.47	0.05
	0.5	0.81	0.81	0.00	0.42	0.47	0.05
	1.0	0.81	0.81	0.00	0.42	0.47	0.05

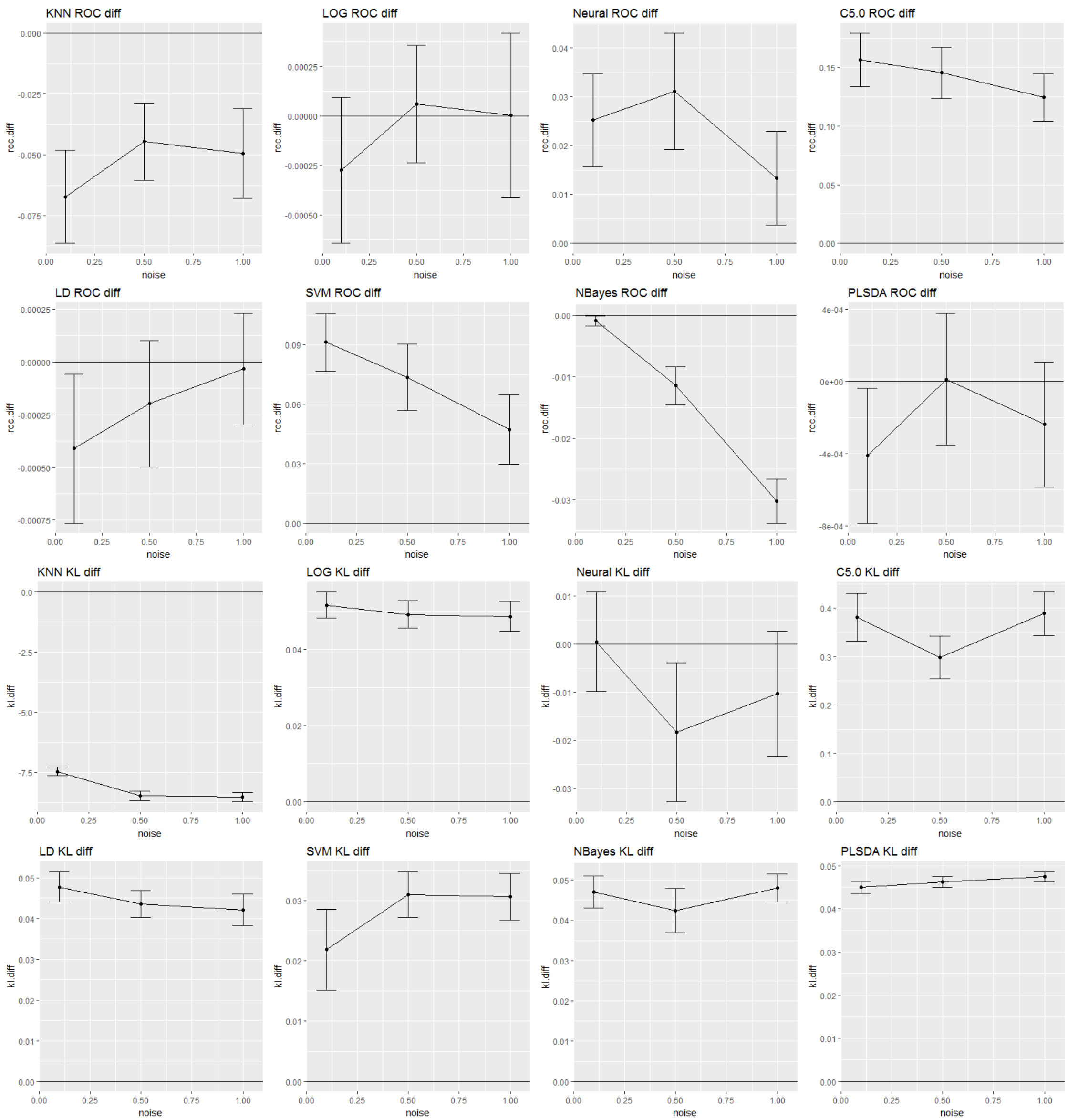


Figure 3.5: Pilot simulation outcome with two-side vibration and `noisy.repl` = 3

Chapter 4

Testing with Real Imbalanced Binary Data Sets

The previous chapter demonstrates the effectiveness of the noisy replication method with a simulated binary data set with IR = 10.0. To further justify our method, more practical validations should be conducted. Furthermore, we are also interested in testing with different imbalance ratios, which could be another potential factor influencing the performance of this innovative noisy replication method. In this thesis, if a data set with a higher IR value, we suppose this data set has a relatively large numerous class and a relatively small rare class.

In this Chapter, we will first introduce real data sets, and then display the outcomes with figures. Meanwhile, the interpretation will be given, focusing on the influence from the noise level, the model selection, and the imbalance ratio.

4.1 Introduction to Real Data Sets

The real data sets are from the website, "Knowledge Extraction based on Evolutionary Learning" (KEEL) [10] and the UC Irvine Machine Learning Repository [22]. These data sets cover a variety of areas, such as glass production, iris study, thyroid disease, breast cancer, etc. Table 4.1 in the end of this chapter shows a brief structure of all data sets studied in this thesis. Each record stands for a data set. *Fea.*, *Real*, *Int.*, and *Nom.* represent the number of all features (instances), and the number of Real/Integer/Nominal valued attributes respectively. *Minor.* is the number of observations in the minority class. *Obs.* is the total number of observations in that data set. *IR* is the imbalance ratio. *Classes* is the number of categories of the data set. This table lists both the simulated and real data sets, as well as both the binary and multi-class imbalanced data sets. The range of IR is relatively broad, varying from 1.86 to 129.44. More specifically, 15 data sets have an IR smaller than 10.0; 18

data sets have an IR between 10.0 and 30.0; 17 data sets have an IR larger than 30.0.

Since the pilot experiment is a success in validating the noisy replication method, we will keep testing with the following parameters: `nsim` = 100, `noisy.repl` = 1, `noisy.train` = 100, `sigma.noise` = 0.1, 0.5, or 1.0, and two-side vibration with eight machine learning models: 10NN, KDA, Logistic, SVM, Neural Network, Naïve Bayes, C5.0, and PLS.

4.2 Results and Interpretations

Table 4.2 summarizes 51 optimal models after adding noise replicates. For each data set, represented by its IR, eight models are listed. The optimal noise (Opt. σ_{noise}) is the noise level where we receive the highest AUC value or the lowest KL distance. For instance, for the data set with IR = 1.86, the AUC value of the 10NN model is 0.39 without applying the noisy replication method. After applying the noisy replication method with $\sigma_{noise} = 1.0$, the AUC value increases to 0.88, and the increase percentage (% inc.) is 123.87%, which is the highest among three noise levels. We call $\sigma_{noise} =$ 1.0 is the *optimal noise* (Opt. σ_{noise}), and the model with the optimal noise is called the *optimal model)*. Also in the same data set, the KL distance decreases the most when $\sigma_{noise} = 0.5$. Examining Table 4.2 intuitively, we can find that after applying the noisy replication method, most models have an improvement assessed by the AUC or KL distance. The higher the IR value is, the better the noisy replication model performs. More analyses based on this table will be provided in the following figures.

Appendix B contains the plots for 95% confident intervals of $\triangle$ROC and $\triangle$KL distance in binary data sets. Each graph has eight small subgraphs representing eight machine learning models after adding noise replicates. The x-axis is the noise level, and the y-axis is $\triangle$ROC (`ROC.diff`). If there is no statistically significant difference between the original model and the basic model and the noisy replication model, then 95% confidence interval will contain $\triangle$ROC = 0. The means of $\triangle$ROC are joined

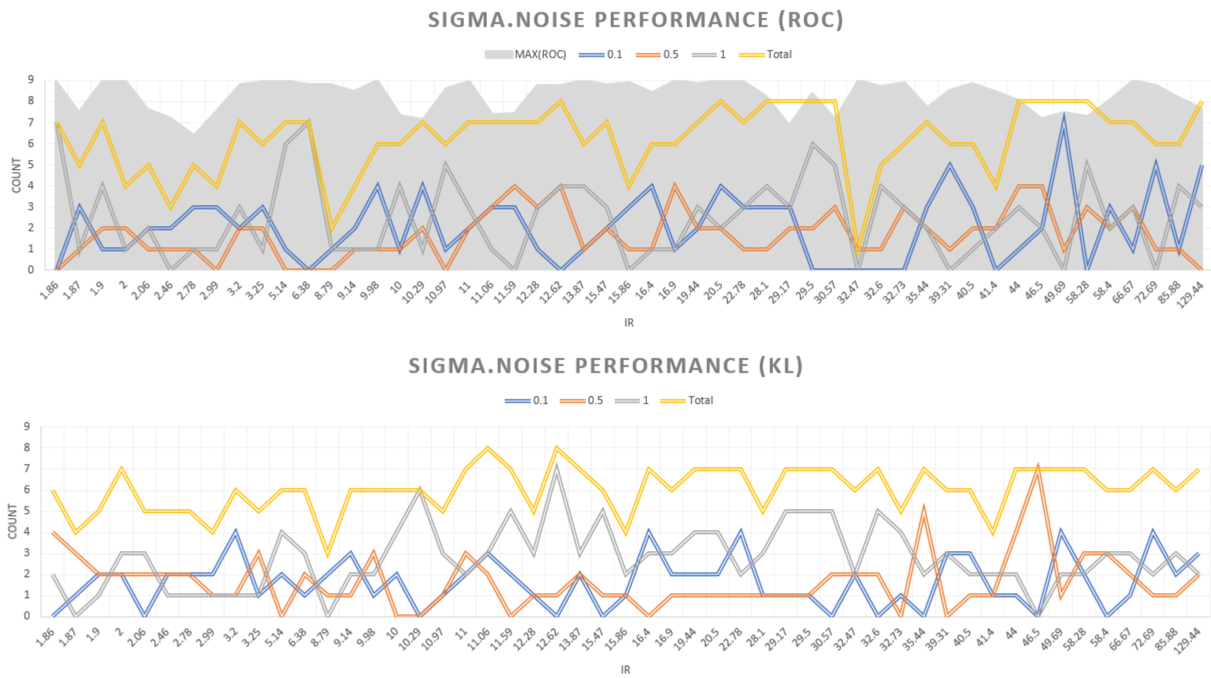


Figure 4.1: Counts of optimal models in each binary data set

by the solid line. For instance, for the data set with IR = 1.86, all three noise levels have a positive increase lying above the x-axis, and the higher the noise level, the more △ROC increases. The optimal model of this model for this particular data set is summarized in Table 4.2. As for △KL distance, if the 95% confidence intervals are below the x-axis, we can tell the model has a good performance. The model with the largest decrease in △KL distance is the optimal model.

From both Table 4.2 and Appendix B, we can tell that most models in many data sets perform as we expected: △ROC above $x = 0$ and △KL distance below $x = 0$. However, the performance of some models may be affected by the noise level and the imbalance ratio. In addition, for some data sets or some models, the noisy replication method does not provide a statistically significant improvement, and even sometimes performs slightly worse than the basic model. This happens more frequently when IR is relatively small. We will continue our discovery in Figure 4.1, Figure 4.2, and Figure 4.3.

Two graphs in Figure 4.1 illustrate the performance of noise levels and the increase

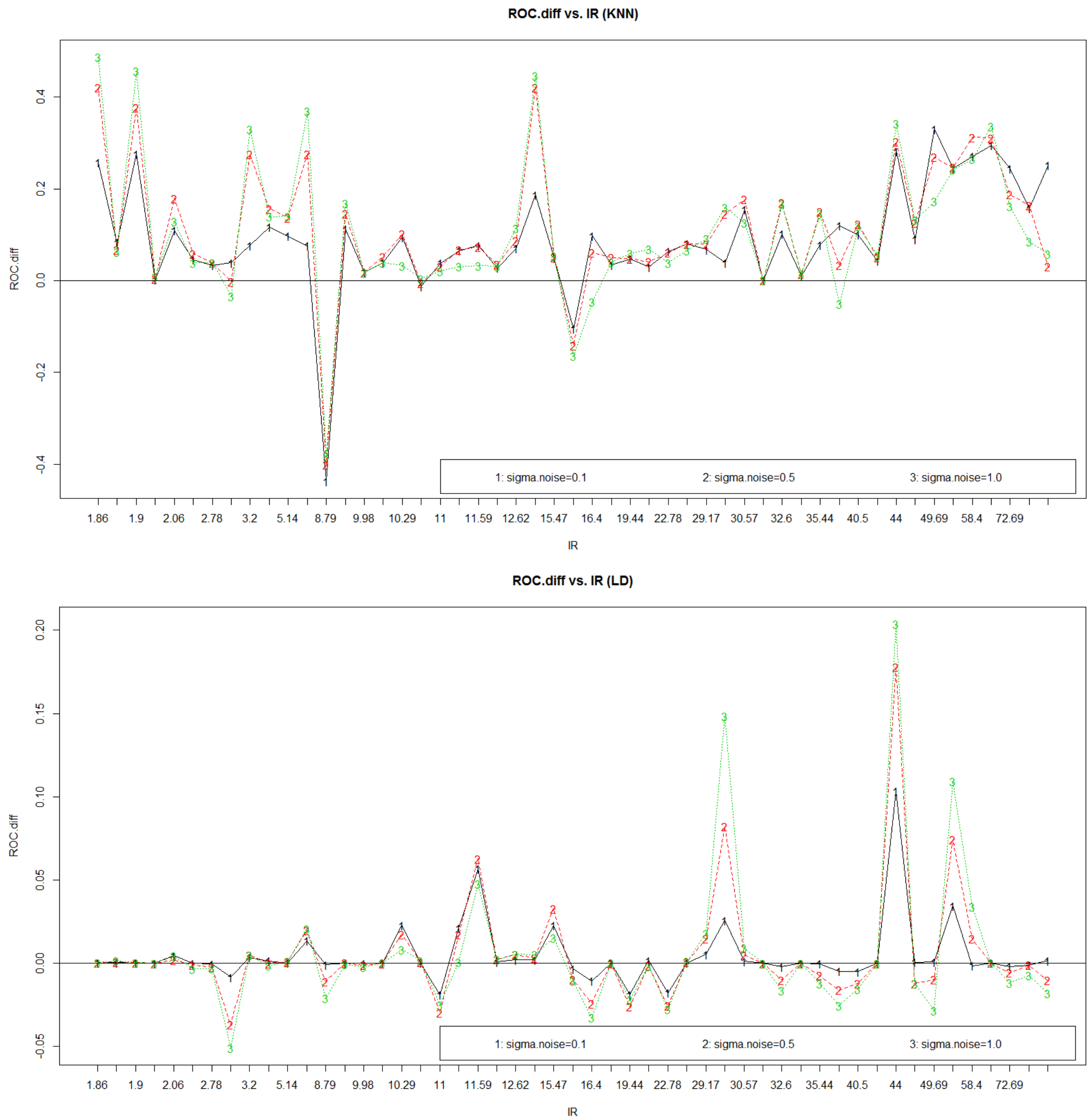


Figure 4.2: $\triangle$ROC vs. IRs in eight models for binary data sets

of imbalance ratio. The x-axis represents each data set by their imbalance ratio values. The left y-axis is the count of optimal models for each data set with different noise levels. The right x-axis in the first graph represents the AUC value of optimal models for each data set. The first graph is assessed by ROC, and the second is by KL distance. The blue line plots the performance when $\sigma_{noise} = 0.1$, the red plots when $\sigma_{noise} = 0.5$, and the gray plots when $\sigma_{noise} = 1.0$. The yellow plot represents the total number of optimal models for all data sets, and its number should be less than or

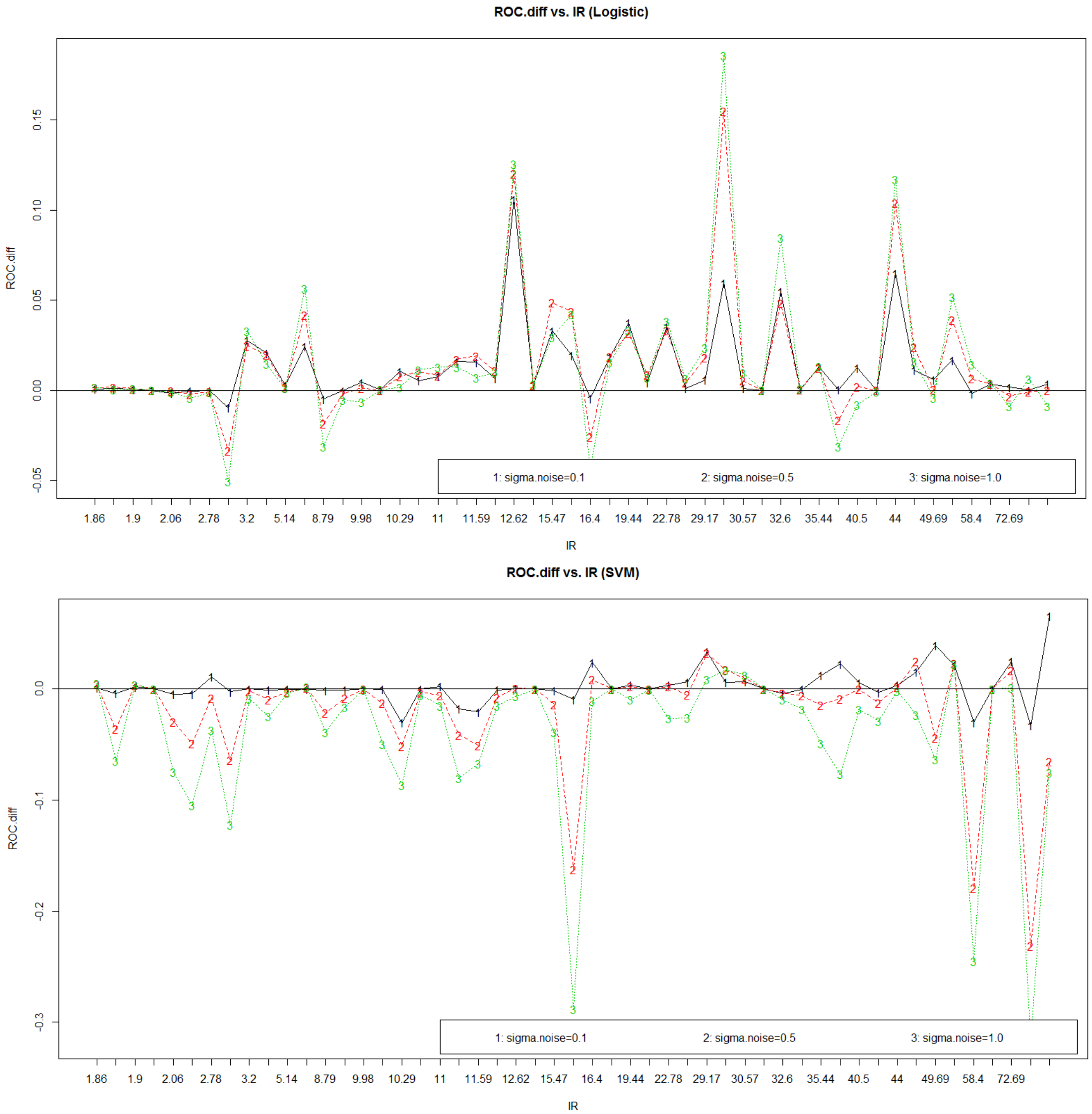
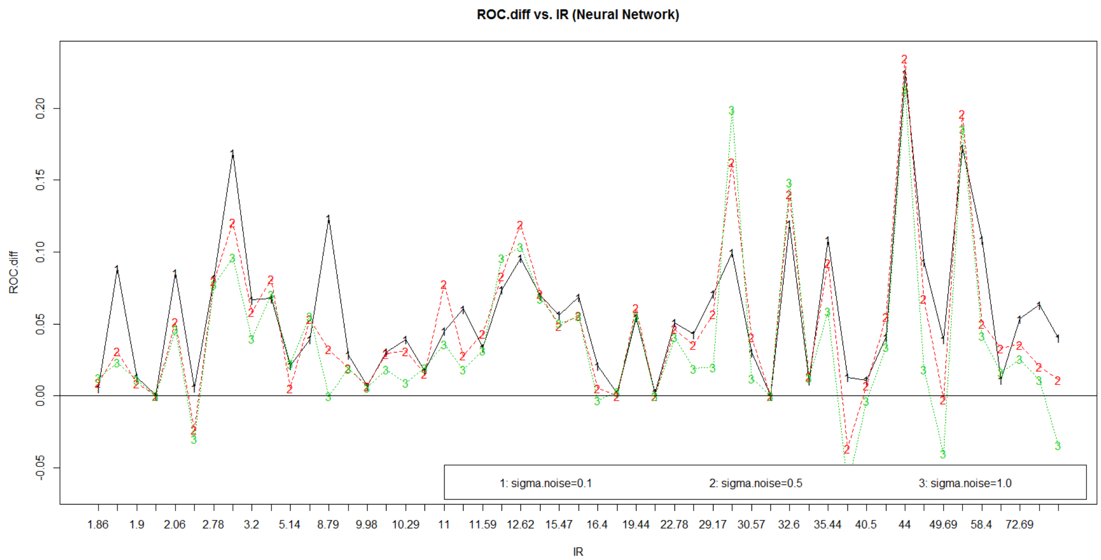


Figure 4.2: △ROC vs. IRs in eight models for binary data sets (cont.)

equal to eight (8), which is the number of machine learning models. The gray shaded area illustrates the best AUC values a data set could get with noisy replicates.

From Figure 4.1, we can infer that there are some similarities between the two graphs and the two assessments: models with $\sigma_{noise} = 0.5$ and $\sigma_{noise} = 1.0$ perform better than the smaller noise level; more than half of the testing models perform better after adding noisy replicates. Meanwhile, we can see from the yellow line that the total counts of optimal models assessed by ROC is not as stable as those assessed by KL distance. However, this does not mean the model itself is not good, or the



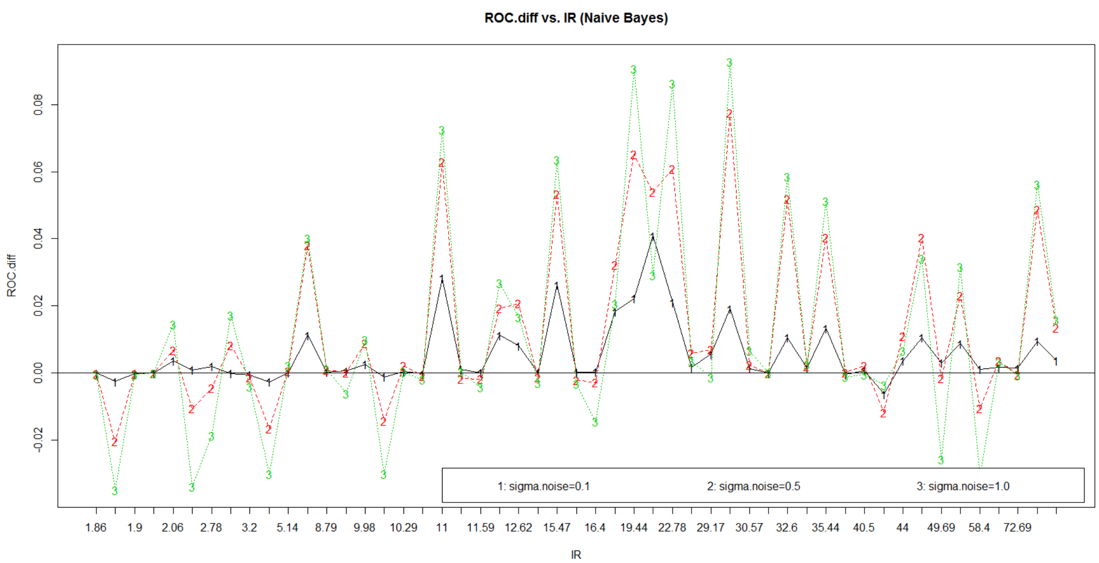


Figure 4.2: △ROC vs. IRs in eight models for binary data sets (cont.)

noisy replication method is not effective, since both the original and improved AUC values are close to 1.0. Unfortunately, Figure 4.1 does not show us a statistically significant connection between the noise level and the imbalance ratio.

Figure 4.2 is another way to illustrate the performance of noise levels and the imbalance ratio, and they are measured by △ROC. Figure 4.2 contains eight graphs in three consecutive pages. Each graph represents the performance for a machine learning model. The x-axis is the imbalance ratio for each data set and the y-axis is

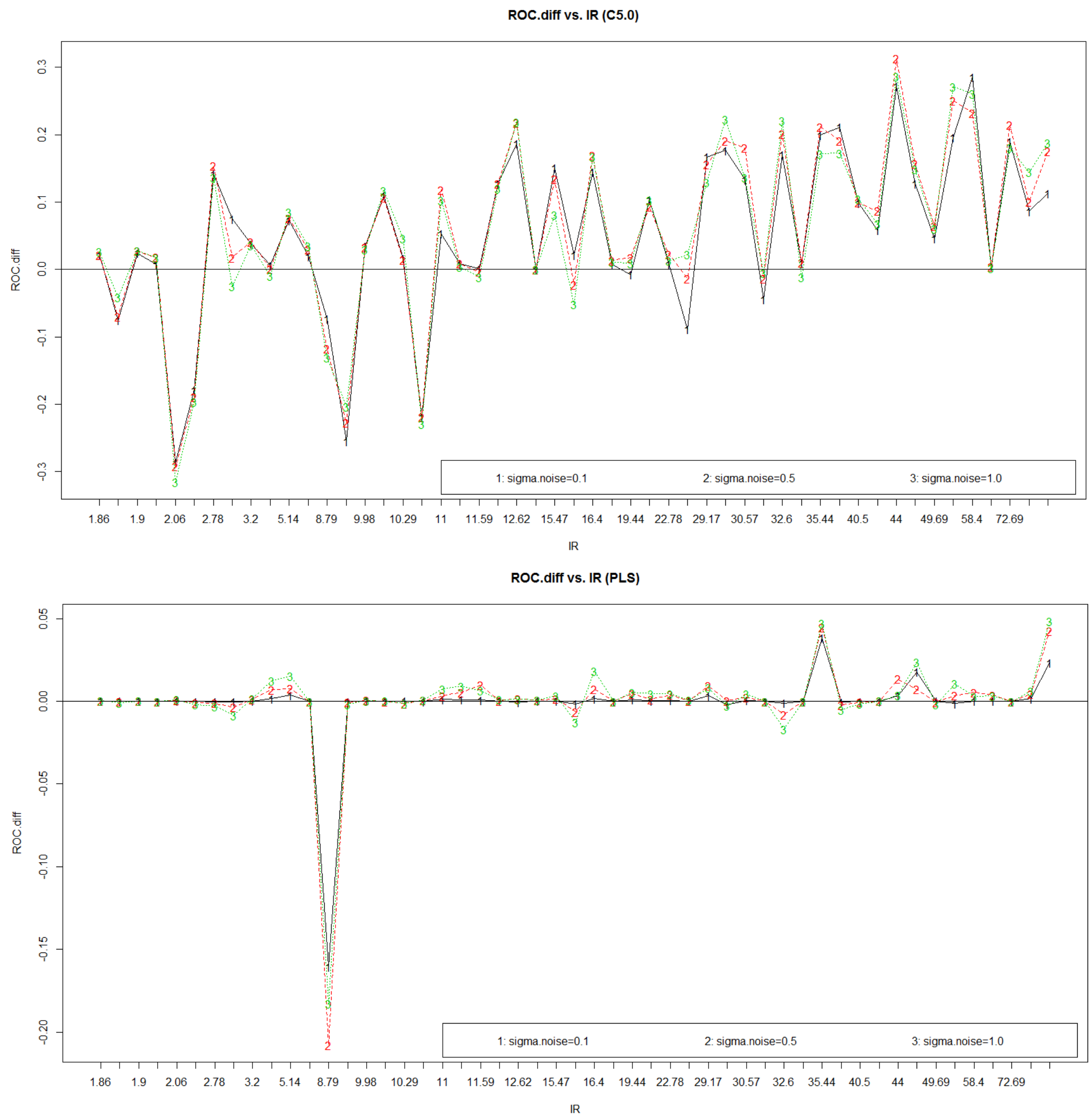


Figure 4.2: $\triangle$ROC vs. IRs in eight models for binary data sets (cont.)

the value of $\triangle$ROC (`ROC.diff`). Three colored dashed lines, black, red, and green, represent the condition when $\sigma_{noise} = 0.1, 0.5$, and 1.0 respectively.

Here is the analysis for eight machine learning models after applying the noisy replication method. The 1st subgraph in Figure 4.2 shows that learning with KNN, all three noise levels have a good performance, since most $\triangle$ROC are larger than 0. We can also see a U-shaped curve above the line of $\triangle$ROC $= 0$, which means the KNN with noisy replication has a better performance when IR is either relatively

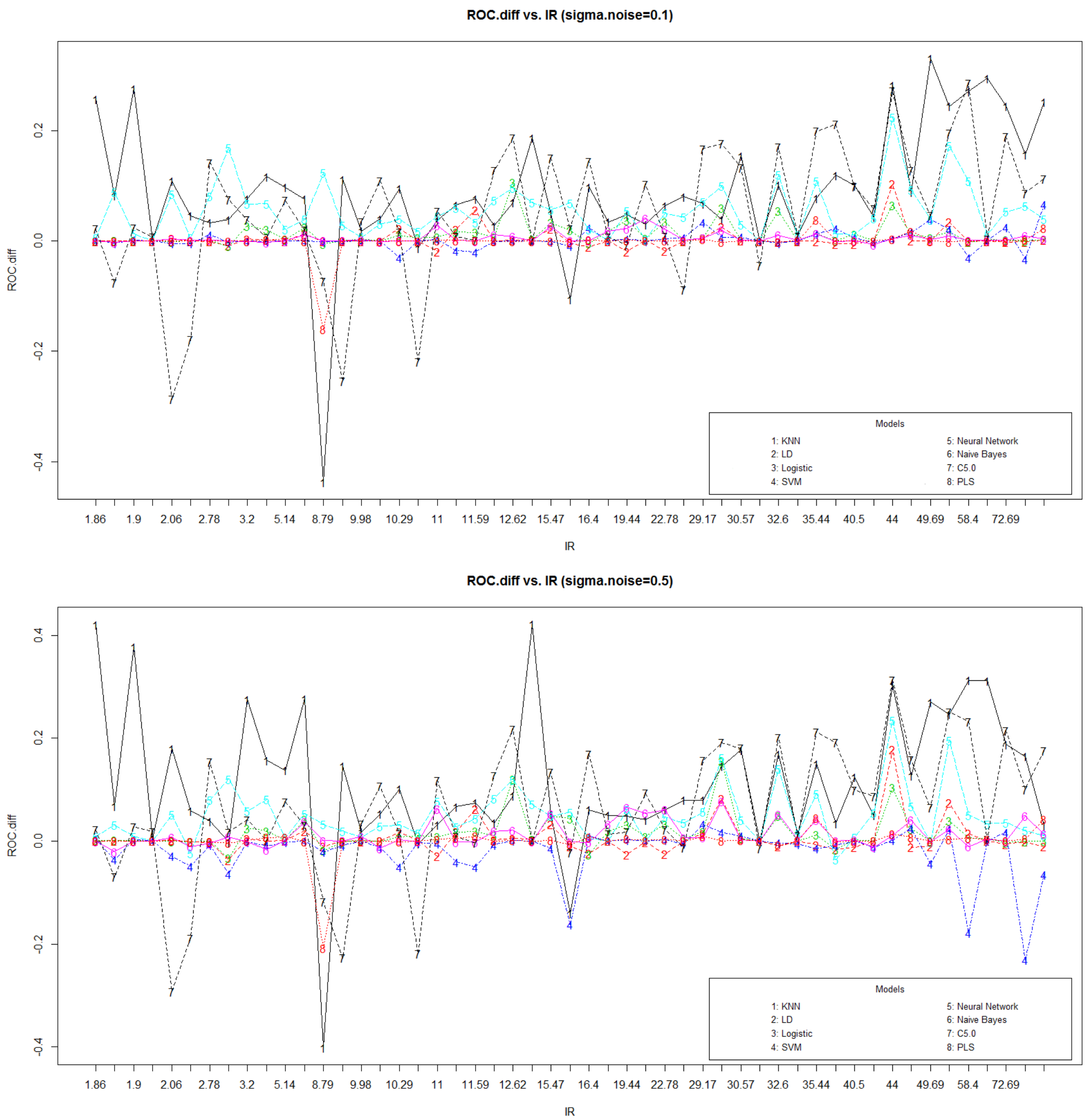


Figure 4.3: Model performance with different noise levels in binary data sets

small or relatively large. As for different noise levels, `sigma.noise` = 0.5 and 1.0 have a better performance when IR is less than 15.0, while all three levels perform similarly when IR is greater than 15.0.

The 2nd subgraph in Figure 4.2 shows that learning with LDA, most $\triangle$ROC values are greater than or close to 0. $\triangle$ROC gets a relatively large variation when IR increases.

The 3rd subgraph in Figure 4.2 shows that learning with the logistic regression,

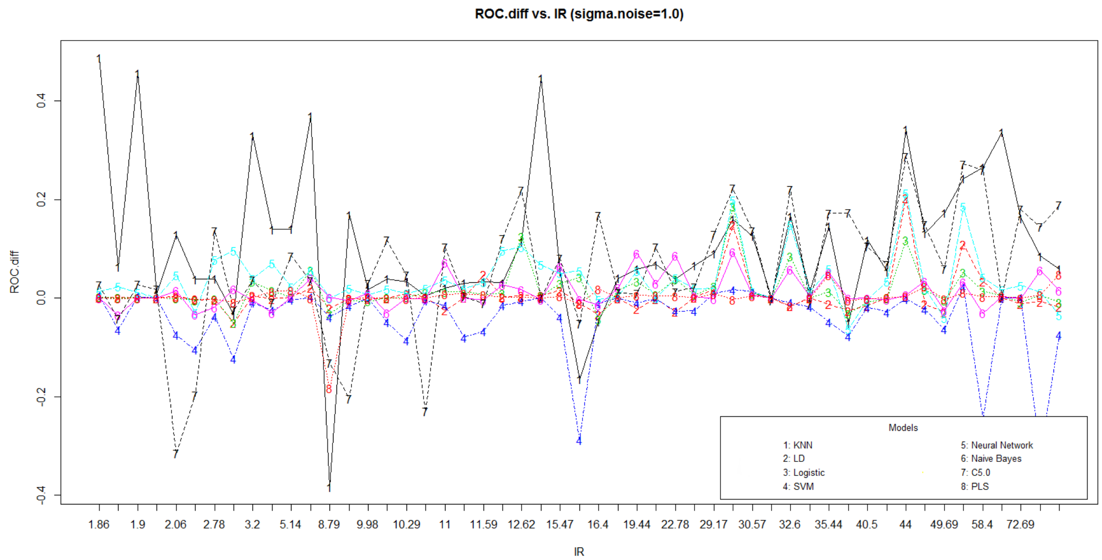


Figure 4.3: Model performance with different noise levels in binary data sets (cont.)

the noisy replication model performs better than in LDA. When IR is between 3.0 and 60.0, we can see most △ROC are greater than or close to 0.

The 4th subgraph in Figure 4.2 shows that learning with SVM, most △ROC are not greater than 0, which means the noisy replication method does not successfully increase the performance.

The 5th subgraph in Figure 4.2 shows that learning with Neural Network, most △ROC are greater than or close to 0 for all three noise levels.

The 6th subgraph in Figure 4.2 shows that learning with Naïve Bayes, we can tell `sigma.noise` = 0.5 and 1.0 lines have a better performance when IR becomes larger than 20.0, and the variation is more significant in ROC than `sigma.noise` = 0.1 line.

The 7th subgraph in Figure 4.2 shows that learning with C5.0, a statistically significant improvement of △ROC comes with the increase of IR, especially when IR becomes larger, as most △ROC are greater than 0.

The 8th subgraph in Figure 4.2 shows that learning with PLS, most △ROC are not greater than 0, which means the noisy replication method does not successfully increase the performance.

Figure 4.3 demonstrates the performance of eight machine learning models along with the increase of imbalance ratio based on the ROC assessment. Each graph represents the condition under one noise level. Different colored lines represent different models. The 1st subgraph in Figure 4.3 shows when `sigma.noise` = 0.1, we can tell that most models have $\triangle$ROC greater than or close to 0. Among all eight models, KNN and neural network perform better; C5.0 has a statistically significant increase of AUC values along with the increase of imbalance ratio.

The 2nd subgraph in Figure 4.3 shows when `sigma.noise` = 0.5, most models have an increase of $\triangle$ROC with the increase of imbalance ratio, especially for linear discriminant analysis, neural network, and C5.0. However, $\triangle$ROC of several models converge to 0 when IR is greater than 50.0, especially for neural network and SVM.

The 3rd subgraph in Figure 4.3 shows when `sigma.noise` = 1.0, we still can observe the relationship between the model performance and the imbalance ratio: the increase of $\triangle$ROC comes with the increase of IR. However, when IR is larger than 50, model performance may drop. Compared among three noise levels, `sigma.noise` = 0.5 and 1.0 is better than `sigma.noise` = 0.1, due to the scale of $\triangle$ROC.

Table 4.1: Data structure for each data set

Real Data Set	Feat.	Real	Int.	Nom.	Minor.	Obs.	IR	Classes
sim(binary)	2	2	0	0	20	220	10.00	2
breast cancer	10	0	10	0	241	698	1.90	2
glass1	9	9	0	0	76	214	1.82	2
wisconsin	9	0	9	0	77	220	1.86	2
pima	8	8	0	0	268	768	1.87	2
iris0	4	4	0	0	50	150	2.00	2
glass0	9	9	0	0	70	214	2.06	2
yeast1	8	8	0	0	429	1484	2.46	2
haberman	3	0	3	0	81	306	2.78	2
vehicle2	18	0	18	0	218	846	2.88	2
vehicle1	18	0	18	0	217	846	2.90	2
vehicle3	18	0	18	0	212	846	2.99	2
glass-0-1-2-3_vs_4-5-6	9	9	0	0	51	214	3.20	2
vehicle0	18	0	18	0	199	846	3.25	2
new-thyroid1	5	4	1	0	35	215	5.14	2
glass6	9	9	0	0	29	214	6.38	2
yeast3	8	8	0	0	163	1484	8.10	2
page-blocks0	10	4	6	0	559	5472	8.79	2
vowel0	13	10	3	0	90	988	9.98	2
glass-0-1-6_vs_2	9	9	0	0	17	192	10.29	2
glass2	9	9	0	0	17	214	11.59	2
shuttle-c0-vs-c4	9	0	9	0	123	1829	13.87	2
yeast-1_vs_7	7	7	0	0	30	459	14.30	2
glass4	9	9	0	0	13	214	15.46	2
ecoli4	7	7	0	0	20	336	15.80	2
page-blocks-1-3_vs_4	10	4	6	0	28	472	15.86	2
abalone9-18	8	7	0	1	42	731	16.40	2
glass-0-1-6_vs_5	9	9	0	0	9	184	19.44	2
shuttle-c2-vs-c4	9	0	9	0	6	129	20.50	2
glass5	9	9	0	0	9	214	22.78	2
yeast-2_vs_8	8	8	0	0	20	482	23.10	2
yeast4	8	8	0	0	51	1484	28.10	2
yeast-1-2-8-9_vs_7	8	8	0	0	30	947	30.57	2
yeast5	8	8	0	0	44	1484	32.73	2
yeast6	8	8	0	0	35	1484	41.40	2
abalone19	8	7	0	1	32	4174	129.44	2
ecoli-0-3-4_vs_5	7	7	0	0	20	200	9.00	2
yeast-0-3-5-9_vs_7-8	8	8	0	0	50	506	9.12	2

Continued on next page

Table 4.1 – continued from previous page

Real Data Set	Feat.	Real	Int.	Nom.	Minor.	Obs.	IR	Classes
yeast-0-2-5-7-9_vs_3-6-8	8	8	0	0	99	1004	9.14	2
ecoli-0-6-7_vs_5	6	6	0	0	20	220	10.00	2
led7digit-0-2-4-5-6-7-8-9_vs_1	7	7	0	0	37	443	10.97	2
glass-0-6_vs_5	9	9	0	0	9	108	11.00	2
glass-0-1-4-6_vs_2	9	9	0	0	17	205	11.06	2
ecoli-0-1-4-7_vs_5-6	6	6	0	0	25	332	12.28	2
cleveland-0_vs_4	13	13	0	0	13	173	12.31	2
dermatology-6	34	0	34	0	20	358	16.90	2
winequality-red-4	11	11	0	0	53	1599	29.17	2
poker-9_vs_7	10	0	10	0	8	244	29.50	2
abalone-3_vs_11	8	7	0	1	15	502	32.47	2
winequality-white-9_vs_4	11	11	0	0	5	168	32.60	2
winequality-red-8_vs_6	11	11	0	0	18	656	35.44	2
abalone-17_vs_7-8-9-10	8	7	0	1	58	2338	39.31	2
abalone-21_vs_8	8	7	0	1	14	581	40.50	2
winequality-white-3_vs_7	11	11	0	0	20	900	44.00	2
winequality-red-8_vs_6-7	11	11	0	0	18	855	46.50	2
abalone-19_vs_10-11-12-13	8	7	0	1	32	1622	49.69	2
winequality-white-3-9_vs_5	11	11	0	0	25	1482	58.28	2
poker-8-9_vs_6	10	0	10	0	25	1485	58.40	2
shuttle-2_vs_5	9	0	9	0	49	3316	66.67	2
winequality-red-3_vs_5	11	11	0	0	10	691	68.10	2
abalone-20_vs_8-9-10	8	7	0	1	26	1916	72.69	2
poker-8_vs_6	10	0	10	0	17	1477	85.88	2
sim(multi)	2	2	0	0	20	400	9.00	3
wine	13	13	0	0	48	178	1.48	3
hayes-roth	4	0	4	0	30	132	1.70	3
penbased	16	16	0	0	105	1100	1.10	10
new-thyroid	5	4	1	0	30	215	5.00	3

Continued on next page

Table 4.1 – continued from previous page

Real Data Set	Feat.	Real	Int.	Nom.	Minor.	Obs.	IR	Classes
balance	4	4	0	0	49	625	5.88	3
glass	9	9	0	0	9	214	8.44	6 (7)
yeast	8	8	0	0	5	1484	92.60	10
ecoli	7	7	0	0	5	336	28.60	8
pageblocks	10	10	0	0	3	548	164.00	5
shuttle	9	0	9	0	2	2175	853.00	5 (7)

Table 4.2: Optimal noise level for each binary data set

IR	Model	Opt. σ_{noise}	ROC/AUC Orig.	ROC/AUC Noisy	% inc.	Opt. σ_{noise}	KL distance Orig.	KL distance Noisy	% dec.
1.86	10NN	1.0	0.39	0.88	123.87%	0.5	141.68	130.72	7.74%
	LDA	1.0	0.99	1.00	0.04%	1.0	0.19	0.12	33.74%
	Logistic	1.0	0.99	1.00	0.17%	0.5	0.24	0.09	62.14%
	SVM	1.0	0.99	0.99	0.48%	0.5	0.10	0.10	3.93%
	Neural	1.0	0.98	0.99	1.27%	0.5	0.99	0.10	90.40%
	NB	0.1	0.99	0.98	-0.01%	0.1	3.94	3.95	-0.41%
	C5.0	1.0	0.97	0.99	2.73%	1.0	0.19	0.19	1.82%
	PLS	1.0	1.00	1.00	0.03%	0.1	0.37	0.37	-0.35%
1.87	10NN	0.1	0.66	0.74	12.49%	0.1	30.24	1.83	93.94%
	LDA	1.0	0.83	0.83	0.16%	0.5	0.49	0.49	0.60%
	Logistic	0.5	0.83	0.83	0.20%	0.5	0.49	0.49	0.64%
	SVM	0.1	0.82	0.82	-0.47%	0.1	0.50	0.50	-0.44%
	Neural	0.1	0.61	0.70	14.45%	0.5	0.69	0.64	7.49%
	NB	0.1	0.81	0.81	-0.31%	0.1	0.66	0.68	-3.45%
	C5.0	1.0	0.74	0.70	-5.66%	0.1	0.61	7.87	-1186.00%
	PLS	0.1	0.80	0.80	0.00%	0.1	0.58	0.58	-0.31%
1.90	10NN	1.0	0.45	0.91	101.30%	0.5	141.36	129.63	8.30%
	LDA	1.0	0.99	0.99	0.03%	1.0	0.16	0.13	17.43%
	Logistic	1.0	0.99	0.99	0.10%	0.5	0.17	0.11	33.79%
	SVM	0.5	0.99	0.99	0.36%	0.1	0.11	0.11	1.27%
	Neural	0.1	0.97	0.99	1.32%	0.1	1.27	0.12	90.49%
	NB	0.1	0.98	0.98	-0.01%	0.1	3.78	3.80	-0.64%
	C5.0	1.0	0.96	0.99	2.88%	0.5	0.21	0.40	-87.62%
	PLS	0.5	0.99	0.99	0.02%	0.1	0.38	0.38	-0.32%
2.00	10NN	1.0	0.50	0.51	1.66%	1.0	153.48	150.98	1.62%
	LDA	0.1	1.00	1.00	0.00%	0.1	0.00	0.00	9.52%
	Logistic	0.5	1.00	1.00	0.00%	0.5	0.03	0.02	10.84%
	SVM	0.1	1.00	1.00	0.00%	1.0	0.03	0.02	25.49%
	Neural	0.1	1.00	1.00	0.05%	0.1	0.01	0.00	74.04%
	NB	0.1	1.00	1.00	0.00%	0.5	0.00	0.00	20.00%
	C5.0	0.5	0.98	1.00	1.83%	1.0	0.07	0.02	74.11%
	PLS	0.1	1.00	1.00	0.00%	0.1	0.34	0.34	-0.11%
2.06	10NN	0.5	0.58	0.76	30.80%	1.0	52.92	30.58	42.21%
	LDA	0.1	0.81	0.81	0.56%	1.0	0.51	0.50	1.90%
	Logistic	0.5	0.81	0.81	-0.06%	0.5	1.05	0.83	21.45%
	SVM	0.1	0.83	0.82	-0.57%	0.1	0.50	0.51	-1.43%
	Neural	0.1	0.76	0.85	11.21%	0.5	1.87	0.47	74.71%
	NB	1.0	0.77	0.79	1.89%	1.0	3.12	2.31	26.01%
	C5.0	0.1	0.82	0.53	-34.98%	0.5	0.57	47.08	-8110.96%

Continued on next page

Table 4.2 – continued from previous page

IR	Model	Opt. σ_{noise}	ROC/AUC Orig.	ROC/AUC Noisy	% inc.	Opt. σ_{noise}	KL distance Orig.	KL distance Noisy	% dec.
	PLS	1.0	0.80	0.80	0.12%	0.1	0.58	0.58	0.00%
2.46	10NN	0.5	0.70	0.76	8.35%	1.0	45.15	2.83	93.74%
	LDA	0.1	0.79	0.79	-0.01%	0.5	0.50	0.50	0.28%
	Logistic	0.1	0.79	0.79	-0.05%	0.1	0.50	0.50	0.05%
	SVM	0.1	0.78	0.77	-0.56%	0.1	0.51	0.51	-1.02%
	Neural	0.1	0.79	0.80	0.70%	0.1	0.52	0.48	8.03%
	NB	0.1	0.76	0.76	0.12%	0.5	3.76	3.74	0.39%
	C5.0	0.1	0.72	0.54	-24.93%	0.1	0.55	57.11	-10190.41%
	PLS	0.1	0.79	0.79	-0.02%	0.1	0.56	0.57	-0.20%
2.78	10NN	1.0	0.65	0.69	5.95%	0.5	25.01	1.75	92.99%
	LDA	0.1	0.67	0.67	-0.06%	0.5	0.56	0.55	1.63%
	Logistic	0.1	0.67	0.67	-0.03%	1.0	0.56	0.55	1.20%
	SVM	0.1	0.69	0.70	1.59%	0.1	0.56	0.55	1.09%
	Neural	0.1	0.63	0.71	12.98%	0.1	0.58	0.53	8.77%
	NB	0.1	0.64	0.64	0.30%	0.1	0.82	0.84	-1.90%
	C5.0	0.5	0.55	0.70	28.01%	0.1	0.59	4.67	-696.06%
	PLS	0.1	0.68	0.68	-0.04%	0.1	0.59	0.59	-0.06%
2.99	10NN	0.1	0.71	0.75	5.44%	0.1	55.31	19.91	64.00%
	LDA	0.1	0.84	0.84	-0.98%	0.1	0.43	0.43	-0.70%
	Logistic	0.1	0.85	0.84	-1.13%	0.1	0.41	0.42	-1.82%
	SVM	0.1	0.83	0.83	-0.29%	0.1	0.45	0.44	0.64%
	Neural	0.1	0.62	0.78	27.45%	0.5	0.61	0.53	12.23%
	NB	1.0	0.70	0.72	2.48%	1.0	1.54	1.42	7.37%
	C5.0	0.1	0.75	0.83	10.00%	0.1	0.63	0.64	-2.76%
	PLS	0.1	0.69	0.69	-0.05%	0.1	0.58	0.58	-0.08%
3.20	10NN	1.0	0.59	0.92	55.55%	0.1	149.30	100.29	32.82%
	LDA	0.5	0.97	0.97	0.51%	1.0	0.36	0.24	32.50%
	Logistic	1.0	0.93	0.97	3.54%	0.1	4.09	0.24	94.11%
	SVM	0.1	0.98	0.98	0.01%	0.1	0.18	0.17	2.67%
	Neural	0.1	0.91	0.97	7.38%	0.1	1.22	0.19	84.78%
	NB	0.1	0.95	0.95	-0.05%	0.5	1.99	1.92	3.49%
	C5.0	0.5	0.92	0.96	4.45%	0.5	0.31	1.36	-341.60%
	PLS	1.0	0.96	0.96	0.17%	0.1	0.40	0.40	-0.41%
3.25	10NN	0.5	0.80	0.95	19.70%	0.5	127.78	97.63	23.59%
	LDA	0.1	0.99	0.99	0.15%	0.1	0.13	0.12	5.78%
	Logistic	0.1	0.97	0.99	2.11%	0.5	4.61	0.12	97.33%
	SVM	0.1	0.99	0.99	-0.09%	0.1	0.10	0.10	-7.59%
	Neural	0.5	0.91	0.99	8.95%	0.5	0.40	0.25	36.65%
	NB	0.1	0.81	0.81	-0.33%	1.0	2.37	1.85	21.98%

Continued on next page

Table 4.2 – continued from previous page

IR	Model	Opt. σ_{noise}	ROC/AUC Orig.	ROC/AUC Noisy	% inc.	Opt. σ_{noise}	KL distance Orig.	KL distance Noisy	% dec.
	C5.0	0.1	0.96	0.97	0.65%	0.1	0.22	0.37	-64.06%
	PLS	1.0	0.89	0.90	1.40%	0.1	0.50	0.50	-0.08%
5.14	10NN	1.0	0.84	0.98	16.60%	1.0	166.34	53.18	68.03%
	LDA	1.0	0.99	0.99	0.10%	1.0	0.17	0.15	13.13%
	Logistic	0.1	1.00	1.00	0.26%	0.1	0.76	0.04	94.83%
	SVM	0.1	1.00	1.00	-0.02%	0.1	0.07	0.07	-9.44%
	Neural	1.0	0.98	1.00	2.30%	0.1	0.10	0.05	53.47%
	NB	1.0	0.99	1.00	0.22%	1.0	0.86	0.60	30.68%
	C5.0	1.0	0.91	1.00	9.34%	1.0	0.22	0.08	63.36%
	PLS	1.0	0.97	0.99	1.57%	0.1	0.40	0.40	-0.27%
6.38	10NN	1.0	0.54	0.91	68.17%	0.1	164.33	120.58	26.62%
	LDA	1.0	0.95	0.97	2.18%	1.0	1.10	0.22	79.64%
	Logistic	1.0	0.89	0.95	6.34%	1.0	5.79	0.29	95.07%
	SVM	1.0	0.98	0.98	0.17%	0.5	0.12	0.11	8.87%
	Neural	1.0	0.91	0.97	6.08%	1.0	0.87	0.13	85.26%
	NB	1.0	0.89	0.93	4.51%	0.5	4.07	2.95	27.35%
	C5.0	1.0	0.93	0.96	3.71%	1.0	0.18	1.77	-861.54%
	PLS	0.1	0.97	0.97	0.00%	0.1	0.38	0.38	-0.12%
8.79	10NN	1.0	0.81	0.43	-47.07%	0.1	188.78	184.51	2.26%
	LDA	0.1	0.92	0.92	-0.09%	0.5	0.23	0.20	14.41%
	Logistic	0.1	0.94	0.93	-0.52%	0.1	0.24	0.17	27.25%
	SVM	0.1	0.98	0.98	-0.11%	0.1	0.11	0.17	-48.31%
	Neural	0.1	0.80	0.92	15.49%	0.1	0.20	0.35	-74.93%
	NB	1.0	0.93	0.93	0.12%	0.1	2.48	2.48	-0.18%
	C5.0	0.1	0.96	0.88	-7.62%	0.1	0.11	5.13	-4652.49%
	PLS	0.1	0.79	0.63	-20.26%	0.1	0.45	0.45	-0.30%
9.14	10NN	1.0	0.76	0.93	22.30%	0.5	168.51	45.75	72.85%
	LDA	0.5	0.94	0.94	0.01%	1.0	0.15	0.13	13.18%
	Logistic	0.1	0.94	0.94	-0.03%	0.1	0.21	0.14	33.39%
	SVM	0.1	0.93	0.93	-0.08%	0.1	0.12	0.12	0.15%
	Neural	0.1	0.91	0.94	3.12%	0.1	0.53	0.12	76.46%
	NB	0.1	0.92	0.92	0.07%	1.0	2.51	2.44	2.74%
	C5.0	1.0	0.85	0.65	-23.89%	1.0	0.18	16.09	-9079.85%
	PLS	0.1	0.94	0.94	-0.02%	0.1	0.40	0.40	-0.15%
9.98	10NN	0.1	0.97	0.99	1.89%	0.5	190.82	150.61	21.07%
	LDA	0.1	0.97	0.97	-0.02%	1.0	0.13	0.12	6.86%
	Logistic	0.1	0.99	0.99	0.44%	0.5	0.70	0.09	87.64%
	SVM	0.1	1.00	1.00	0.00%	0.1	0.02	0.02	-3.60%
	Neural	0.1	0.99	1.00	0.69%	0.5	0.15	0.05	67.58%
	NB	1.0	0.98	0.99	1.01%	1.0	0.15	0.08	42.05%

Continued on next page

Table 4.2 – continued from previous page

IR	Model	Opt. σ_{noise}	ROC/AUC Orig.	ROC/AUC Noisy	% inc.	Opt. σ_{noise}	KL distance Orig.	KL distance Noisy	% dec.
	C5.0	0.1	0.96	1.00	3.59%	0.1	0.10	0.05	54.04%
	PLS	0.5	0.96	0.96	0.07%	0.1	0.40	0.40	-0.05%
10.00	10NN	0.5	0.74	0.79	7.07%	1.0	10.69	2.86	73.28%
	LDA	1.0	0.81	0.81	0.02%	1.0	0.24	0.24	0.36%
	Logistic	1.0	0.80	0.80	0.03%	1.0	0.25	0.24	1.23%
	SVM	0.1	0.67	0.67	-0.03%	0.1	0.27	0.27	2.78%
	Neural	0.1	0.78	0.81	3.91%	1.0	0.32	0.25	22.53%
	NB	0.1	0.81	0.81	-0.14%	0.1	0.24	0.24	0.78%
	C5.0	1.0	0.60	0.72	19.54%	1.0	0.29	0.33	-13.61%
	PLS	1.0	0.81	0.81	0.01%	0.1	0.42	0.42	-0.02%
10.29	10NN	0.5	0.65	0.75	15.55%	1.0	105.24	32.34	69.27%
	LDA	0.1	0.77	0.79	2.95%	1.0	0.35	0.31	10.39%
	Logistic	0.1	0.56	0.57	1.90%	1.0	0.95	0.46	52.19%
	SVM	0.1	0.73	0.70	-4.19%	0.1	0.30	0.30	-0.19%
	Neural	0.1	0.65	0.69	6.00%	1.0	1.65	0.32	80.86%
	NB	0.5	0.67	0.67	0.33%	1.0	2.90	1.81	37.72%
	C5.0	1.0	0.51	0.56	8.94%	1.0	0.35	0.36	-3.98%
	PLS	0.1	0.69	0.69	0.03%	1.0	0.44	0.44	0.10%
10.97	10NN	1.0	0.93	0.93	0.46%	1.0	165.39	157.20	4.95%
	LDA	1.0	0.95	0.95	0.17%	1.0	0.13	0.13	2.70%
	Logistic	1.0	0.93	0.94	1.22%	1.0	1.06	0.38	64.53%
	SVM	0.1	0.93	0.93	0.00%	0.1	0.14	0.14	-0.88%
	Neural	1.0	0.93	0.95	2.10%	0.1	0.70	0.15	78.14%
	NB	0.1	0.95	0.95	-0.03%	0.5	0.78	0.76	3.20%
	C5.0	0.1	0.92	0.70	-23.94%	0.5	0.15	11.36	-7400.57%
	PLS	1.0	0.96	0.96	0.09%	0.1	0.39	0.39	-0.01%
11.00	10NN	0.1	0.89	0.93	4.23%	0.5	187.71	102.30	45.50%
	LDA	0.1	0.93	0.91	-2.05%	0.5	0.48	0.30	36.52%
	Logistic	1.0	0.98	1.00	1.31%	1.0	0.72	0.07	90.82%
	SVM	0.1	0.97	0.98	0.25%	0.1	0.11	0.11	-2.62%
	Neural	0.5	0.91	0.99	8.48%	0.1	0.16	0.08	50.89%
	NB	1.0	0.85	0.92	8.57%	1.0	3.57	0.73	79.62%
	C5.0	0.5	0.86	0.98	13.69%	0.5	0.17	0.11	35.58%
	PLS	1.0	0.88	0.89	0.86%	0.1	0.41	0.41	0.02%
11.06	10NN	0.5	0.69	0.76	9.57%	0.5	114.18	44.84	60.73%
	LDA	0.1	0.80	0.82	2.64%	0.5	0.31	0.28	7.53%
	Logistic	0.5	0.57	0.58	3.02%	0.1	0.89	0.46	48.17%
	SVM	0.1	0.73	0.71	-2.42%	0.1	0.29	0.29	1.09%
	Neural	0.1	0.66	0.72	9.21%	0.1	2.19	0.29	86.85%
	NB	0.1	0.70	0.70	0.14%	1.0	3.36	2.14	36.35%

Continued on next page

Table 4.2 – continued from previous page

IR	Model	Opt. σ_{noise}	ROC/AUC Orig.	ROC/AUC Noisy	% inc.	Opt. σ_{noise}	KL distance Orig.	KL distance Noisy	% dec.
	C5.0	0.5	0.53	0.54	1.61%	1.0	0.35	0.35	0.40%
	PLS	1.0	0.70	0.71	1.31%	1.0	0.43	0.43	0.10%
11.59	10NN	0.1	0.71	0.79	10.73%	1.0	116.11	47.61	58.99%
	LDA	0.5	0.76	0.82	8.25%	0.1	0.31	0.26	13.92%
	Logistic	0.5	0.59	0.61	3.19%	1.0	0.88	0.34	61.58%
	SVM	0.1	0.72	0.70	-2.83%	0.1	0.28	0.28	0.78%
	Neural	0.5	0.67	0.71	6.46%	1.0	1.81	0.28	84.69%
	NB	0.1	0.71	0.71	0.01%	1.0	2.86	1.93	32.55%
	C5.0	0.1	0.55	0.55	0.17%	1.0	0.33	1.22	-270.71%
	PLS	0.5	0.71	0.72	1.39%	1.0	0.42	0.42	0.09%
12.28	10NN	0.5	0.93	0.96	3.76%	1.0	184.20	100.08	45.67%
	LDA	0.5	0.94	0.94	0.24%	1.0	0.19	0.12	35.58%
	Logistic	0.5	0.93	0.94	1.15%	0.5	1.61	0.12	92.33%
	SVM	0.1	0.97	0.97	-0.10%	0.1	0.08	0.08	-3.12%
	Neural	1.0	0.86	0.96	11.07%	0.1	0.71	0.12	83.25%
	NB	1.0	0.93	0.95	2.89%	1.0	0.85	0.77	9.58%
	C5.0	0.1	0.80	0.93	15.94%	1.0	0.19	2.13	-1022.82%
	PLS	1.0	0.94	0.94	0.10%	0.1	0.38	0.38	-0.05%
12.62	10NN	1.0	0.60	0.72	19.00%	1.0	132.85	9.59	92.78%
	LDA	1.0	0.97	0.97	0.56%	1.0	0.23	0.13	42.62%
	Logistic	1.0	0.82	0.95	15.26%	1.0	10.40	0.23	97.79%
	SVM	0.5	0.97	0.97	0.08%	0.5	0.14	0.14	6.05%
	Neural	0.5	0.80	0.92	14.90%	1.0	0.42	0.18	57.46%
	NB	0.5	0.92	0.94	2.27%	1.0	1.19	0.51	57.08%
	C5.0	1.0	0.73	0.95	29.91%	1.0	0.29	0.22	23.99%
	PLS	0.5	0.72	0.72	0.23%	1.0	0.42	0.41	0.15%
13.87	10NN	1.0	0.49	0.94	90.91%	0.5	214.65	213.98	0.31%
	LDA	1.0	0.99	0.99	0.51%	1.0	0.22	0.08	63.39%
	Logistic	1.0	1.00	1.00	0.33%	1.0	0.15	0.00	98.87%
	SVM	0.1	1.00	1.00	0.00%	0.1	0.01	0.01	17.96%
	Neural	0.5	0.93	1.00	7.67%	1.0	0.04	0.01	83.62%
	NB	0.1	1.00	1.00	0.00%	0.1	0.11	0.09	15.10%
	C5.0	0.1	1.00	1.00	0.00%	0.5	0.00	0.00	100.00%
	PLS	1.0	0.99	0.99	0.07%	0.1	0.34	0.34	-0.37%
15.47	10NN	1.0	0.89	0.95	6.02%	1.0	195.33	164.20	15.94%
	LDA	0.5	0.90	0.93	3.63%	1.0	0.41	0.21	47.33%
	Logistic	0.5	0.87	0.92	5.60%	0.5	6.59	0.45	93.17%
	SVM	0.1	0.98	0.98	-0.14%	0.1	0.11	0.11	-4.16%
	Neural	0.1	0.92	0.98	6.08%	1.0	0.31	0.13	58.04%
	NB	1.0	0.75	0.81	8.47%	1.0	2.47	0.64	74.02%

Continued on next page

Table 4.2 – continued from previous page

IR	Model	Opt. σ_{noise}	ROC/AUC Orig.	ROC/AUC Noisy	% inc.	Opt. σ_{noise}	KL distance Orig.	KL distance Noisy	% dec.
	C5.0	0.1	0.76	0.91	19.78%	0.5	0.24	0.60	-153.08%
	PLS	1.0	0.90	0.91	0.36%	1.0	0.38	0.38	0.05%
15.86	10NN	0.1	0.94	0.84	-11.20%	0.1	192.07	184.92	3.73%
	LDA	0.1	0.96	0.96	-0.35%	1.0	0.71	0.21	69.92%
	Logistic	0.5	0.93	0.98	4.69%	1.0	5.38	0.10	98.13%
	SVM	0.1	0.99	0.98	-0.95%	0.1	0.10	0.11	-17.92%
	Neural	0.1	0.90	0.97	7.61%	0.5	0.19	0.20	-5.42%
	NB	0.1	0.95	0.95	0.02%	0.5	3.40	3.18	6.34%
	C5.0	0.1	0.97	0.99	2.25%	0.1	0.05	0.07	-40.85%
	PLS	0.1	0.95	0.95	-0.17%	0.1	0.37	0.37	-0.30%
16.40	10NN	0.1	0.72	0.81	13.64%	0.1	169.30	95.56	43.56%
	LDA	0.1	0.95	0.94	-1.12%	0.1	0.16	0.14	13.68%
	Logistic	0.1	0.94	0.94	-0.48%	0.1	0.13	0.12	7.51%
	SVM	0.1	0.83	0.85	2.88%	0.1	0.17	0.17	4.14%
	Neural	0.1	0.90	0.92	2.31%	1.0	0.51	0.17	66.39%
	NB	0.1	0.75	0.75	0.03%	1.0	0.58	0.48	17.03%
	C5.0	0.5	0.62	0.79	27.16%	0.5	0.21	2.82	-1215.04%
	PLS	1.0	0.75	0.77	2.45%	1.0	0.39	0.39	0.21%
16.90	10NN	0.5	0.92	0.97	5.45%	1.0	207.16	178.14	14.01%
	LDA	0.1	1.00	1.00	0.00%	0.1	0.08	0.02	74.45%
	Logistic	0.5	0.98	1.00	1.91%	0.1	1.17	0.03	97.37%
	SVM	0.1	1.00	1.00	0.00%	1.0	0.01	0.01	28.21%
	Neural	1.0	1.00	1.00	0.30%	1.0	0.01	0.00	64.00%
	NB	0.5	0.92	0.96	3.48%	0.5	36.54	22.30	38.98%
	C5.0	0.5	0.98	0.99	1.39%	1.0	0.03	0.54	-1701.46%
	PLS	0.1	1.00	1.00	0.00%	0.1	0.35	0.35	-0.02%
19.44	10NN	1.0	0.84	0.90	7.00%	1.0	189.46	135.57	28.45%
	LDA	0.1	0.93	0.91	-2.01%	1.0	0.23	0.20	16.34%
	Logistic	0.1	0.95	0.98	3.94%	0.5	3.81	0.07	98.25%
	SVM	0.1	0.96	0.97	0.37%	0.1	0.13	0.12	9.80%
	Neural	0.5	0.92	0.98	6.62%	0.1	0.19	0.09	54.53%
	NB	1.0	0.84	0.93	10.84%	1.0	2.24	0.39	82.46%
	C5.0	0.5	0.96	0.98	1.82%	1.0	0.10	0.18	-80.24%
	PLS	1.0	0.91	0.91	0.59%	1.0	0.38	0.38	0.21%
20.50	10NN	1.0	0.91	0.98	7.49%	1.0	219.64	215.09	2.07%
	LDA	0.1	0.99	0.99	0.17%	1.0	2.91	0.28	90.44%
	Logistic	0.5	0.99	1.00	0.87%	1.0	2.22	0.02	98.96%
	SVM	0.1	1.00	1.00	0.01%	0.1	0.06	0.05	3.01%
	Neural	0.1	1.00	1.00	0.23%	0.1	0.00	0.00	56.24%
	NB	0.5	0.95	1.00	5.71%	0.5	2.20	0.07	97.00%

Continued on next page

Table 4.2 – continued from previous page

IR	Model	Opt. σ_{noise}	ROC/AUC Orig.	ROC/AUC Noisy	% inc.	Opt. σ_{noise}	KL distance Orig.	KL distance Noisy	% dec.
	C5.0	0.1	0.85	0.96	12.07%	1.0	0.12	0.78	-561.36%
	PLS	1.0	0.94	0.94	0.49%	1.0	0.37	0.37	0.07%
22.78	10NN	0.1	0.86	0.92	7.45%	1.0	189.33	128.60	32.08%
	LDA	0.1	0.93	0.91	-1.91%	0.1	0.18	0.17	4.95%
	Logistic	1.0	0.95	0.99	4.02%	0.1	3.09	0.10	96.64%
	SVM	0.1	0.96	0.96	0.38%	0.1	0.12	0.10	9.28%
	Neural	0.1	0.94	0.99	5.45%	0.1	0.19	0.07	64.67%
	NB	1.0	0.84	0.92	10.29%	0.5	2.93	0.91	69.10%
	C5.0	0.5	0.95	0.98	2.37%	1.0	0.09	0.15	-57.10%
	PLS	1.0	0.91	0.91	0.49%	1.0	0.37	0.37	0.14%
28.10	10NN	0.1	0.84	0.92	9.59%	1.0	193.27	59.03	69.46%
	LDA	1.0	0.87	0.88	0.11%	0.5	0.12	0.11	3.46%
	Logistic	1.0	0.86	0.87	0.74%	1.0	0.11	0.11	1.25%
	SVM	0.1	0.84	0.85	0.77%	0.1	0.12	0.12	0.65%
	Neural	0.1	0.86	0.90	5.02%	1.0	0.26	0.11	57.47%
	NB	0.5	0.84	0.85	0.71%	0.1	4.81	4.83	-0.46%
	C5.0	1.0	0.71	0.73	3.14%	1.0	0.13	3.61	-2598.38%
	PLS	1.0	0.88	0.88	0.07%	0.1	0.36	0.36	0.00%
29.17	10NN	1.0	0.47	0.56	19.34%	0.5	164.27	9.32	94.32%
	LDA	1.0	0.75	0.76	2.39%	1.0	0.15	0.14	5.04%
	Logistic	1.0	0.72	0.74	3.27%	1.0	0.24	0.14	42.06%
	SVM	0.1	0.67	0.70	5.03%	0.1	0.14	0.14	1.33%
	Neural	0.1	0.66	0.73	10.71%	1.0	0.34	0.15	56.31%
	NB	0.5	0.69	0.70	1.01%	1.0	0.63	0.53	15.91%
	C5.0	0.1	0.52	0.68	32.30%	0.1	0.15	0.29	-91.59%
	PLS	0.5	0.61	0.62	1.51%	1.0	0.36	0.36	0.01%
29.50	10NN	1.0	0.66	0.82	24.23%	1.0	202.87	145.85	28.11%
	LDA	1.0	0.63	0.78	23.37%	1.0	0.32	0.19	41.75%
	Logistic	1.0	0.63	0.81	29.69%	1.0	7.19	0.28	96.15%
	SVM	0.5	0.91	0.93	1.88%	1.0	0.10	0.08	15.80%
	Neural	1.0	0.60	0.80	33.15%	0.1	1.16	0.16	86.56%
	NB	1.0	0.62	0.71	15.05%	0.5	1.29	0.36	72.27%
	C5.0	1.0	0.57	0.79	39.31%	1.0	0.17	0.38	-131.37%
	PLS	0.5	0.62	0.62	0.06%	1.0	0.36	0.36	0.34%
30.57	10NN	0.5	0.53	0.71	33.21%	1.0	174.66	22.77	86.96%
	LDA	1.0	0.77	0.78	1.18%	1.0	0.12	0.12	1.90%
	Logistic	1.0	0.78	0.79	1.25%	1.0	0.12	0.12	2.71%
	SVM	1.0	0.68	0.70	1.85%	0.5	0.12	0.12	2.53%
	Neural	0.5	0.73	0.77	5.60%	0.5	0.38	0.12	68.54%
	NB	1.0	0.74	0.74	0.90%	1.0	6.45	5.88	8.97%

Continued on next page

Table 4.2 – continued from previous page

IR	Model	Opt. σ_{noise}	ROC/AUC Orig.	ROC/AUC Noisy	% inc.	Opt. σ_{noise}	KL distance Orig.	KL distance Noisy	% dec.
	C5.0	0.5	0.52	0.70	34.77%	0.5	0.14	0.23	-60.33%
	PLS	1.0	0.76	0.77	0.57%	1.0	0.36	0.36	0.01%
32.47	10NN	0.5	1.00	1.00	0.03%	1.0	221.04	220.69	0.16%
	LDA	0.1	1.00	1.00	0.00%	0.5	0.00	0.00	15.77%
	Logistic	0.1	1.00	1.00	0.00%	0.1	0.00	0.00	3.18%
	SVM	0.1	1.00	1.00	0.00%	1.0	0.01	0.01	6.99%
	Neural	0.1	1.00	1.00	0.00%	0.1	0.00	0.00	0.70%
	NB	0.1	1.00	1.00	0.00%	0.5	0.01	0.00	89.28%
	C5.0	1.0	1.00	0.99	-0.51%	1.0	0.03	0.11	-306.36%
	PLS	0.1	1.00	1.00	0.00%	0.1	0.33	0.33	0.00%
32.60	10NN	0.5	0.64	0.81	26.50%	1.0	187.77	91.86	51.08%
	LDA	0.1	0.97	0.97	-0.22%	0.5	0.12	0.11	7.91%
	Logistic	1.0	0.75	0.84	11.21%	1.0	5.85	0.84	85.63%
	SVM	0.5	0.94	0.93	-0.40%	0.5	0.11	0.10	8.38%
	Neural	1.0	0.61	0.76	24.38%	1.0	0.46	0.14	68.46%
	NB	1.0	0.75	0.80	7.84%	1.0	3.20	0.69	78.45%
	C5.0	1.0	0.65	0.87	33.89%	1.0	0.17	1.52	-796.86%
	PLS	0.1	0.74	0.74	-0.14%	1.0	0.36	0.36	0.03%
32.73	10NN	0.5	0.97	0.99	1.43%	1.0	210.74	180.79	14.21%
	LDA	0.5	0.99	0.99	0.00%	1.0	0.06	0.06	2.31%
	Logistic	1.0	0.99	0.99	0.08%	1.0	0.05	0.05	5.08%
	SVM	0.1	0.98	0.98	-0.01%	0.1	0.06	0.06	-0.06%
	Neural	0.5	0.98	0.99	1.39%	1.0	0.20	0.05	77.23%
	NB	1.0	0.96	0.97	0.25%	0.1	2.74	2.73	0.28%
	C5.0	0.5	0.94	0.95	1.04%	0.5	0.07	0.41	-511.82%
	PLS	1.0	0.99	0.99	0.00%	0.1	0.35	0.35	0.00%
35.44	10NN	0.5	0.44	0.59	33.83%	0.5	176.22	11.25	93.61%
	LDA	0.1	0.86	0.86	-0.02%	0.5	0.12	0.11	6.32%
	Logistic	0.1	0.84	0.86	1.55%	0.5	0.15	0.11	26.42%
	SVM	0.1	0.74	0.76	1.67%	0.5	0.12	0.12	1.67%
	Neural	0.1	0.72	0.82	15.16%	1.0	0.65	0.12	81.26%
	NB	1.0	0.72	0.77	7.14%	0.5	0.38	0.22	43.27%
	C5.0	0.5	0.53	0.74	39.77%	1.0	0.13	0.18	-33.91%
	PLS	1.0	0.58	0.63	8.04%	1.0	0.35	0.35	0.03%
39.31	10NN	0.1	0.80	0.92	14.86%	0.1	206.34	110.70	46.35%
	LDA	0.1	0.95	0.95	-0.51%	1.0	0.13	0.08	35.19%
	Logistic	0.1	0.94	0.94	0.03%	1.0	0.14	0.08	42.97%
	SVM	0.1	0.82	0.84	2.76%	0.1	0.10	0.10	7.80%
	Neural	0.1	0.92	0.93	1.41%	0.1	0.11	0.07	29.54%
	NB	0.5	0.78	0.78	0.03%	1.0	0.60	0.51	15.71%

Continued on next page

Table 4.2 – continued from previous page

IR	Model	Opt. σ_{noise}	ROC/AUC Orig.	ROC/AUC Noisy	% inc.	Opt. σ_{noise}	KL distance Orig.	KL distance Noisy	% dec.
	C5.0	0.1	0.60	0.81	35.20%	0.1	0.12	0.26	-129.24%
	PLS	0.1	0.95	0.95	-0.04%	0.1	0.34	0.34	-0.04%
40.50	10NN	0.5	0.79	0.92	15.58%	0.1	213.88	152.28	28.80%
	LDA	0.1	0.98	0.98	-0.52%	1.0	0.12	0.06	48.91%
	Logistic	0.1	0.96	0.97	1.31%	0.1	0.62	0.05	92.60%
	SVM	0.1	0.90	0.90	0.65%	0.5	0.06	0.06	4.74%
	Neural	0.1	0.95	0.96	1.16%	0.1	0.38	0.05	85.86%
	NB	0.5	0.88	0.88	0.25%	1.0	0.62	0.40	35.89%
	C5.0	1.0	0.77	0.87	13.54%	1.0	0.09	0.84	-879.20%
	PLS	0.1	0.98	0.98	-0.01%	0.1	0.33	0.33	-0.07%
41.40	10NN	1.0	0.88	0.93	6.14%	1.0	202.36	89.14	55.95%
	LDA	0.1	0.94	0.94	-0.01%	1.0	0.08	0.07	3.17%
	Logistic	0.1	0.94	0.94	0.00%	0.5	0.07	0.07	0.60%
	SVM	0.1	0.85	0.85	-0.36%	0.1	0.07	0.07	-0.21%
	Neural	0.5	0.89	0.94	6.20%	0.1	0.28	0.07	75.60%
	NB	1.0	0.92	0.91	-0.39%	1.0	4.07	6.13	-50.42%
	C5.0	0.5	0.74	0.83	11.68%	1.0	0.09	1.17	-1152.50%
	PLS	1.0	0.93	0.93	0.02%	0.1	0.34	0.34	0.00%
44.00	10NN	1.0	0.51	0.86	66.62%	0.1	202.92	24.64	87.86%
	LDA	1.0	0.64	0.84	31.80%	0.5	0.16	0.10	39.21%
	Logistic	1.0	0.72	0.84	16.21%	0.5	0.16	0.08	48.08%
	SVM	0.1	0.87	0.88	0.42%	0.5	0.07	0.07	7.50%
	Neural	0.5	0.66	0.89	35.62%	0.5	0.20	0.08	61.72%
	NB	0.5	0.87	0.88	1.26%	1.0	0.27	0.19	31.54%
	C5.0	0.5	0.57	0.88	54.97%	0.5	0.11	0.44	-296.30%
	PLS	0.5	0.58	0.59	2.36%	1.0	0.35	0.35	0.16%
46.50	10NN	1.0	0.46	0.59	28.91%	0.5	187.63	15.37	91.81%
	LDA	0.1	0.79	0.79	0.08%	0.5	0.10	0.10	1.95%
	Logistic	0.5	0.78	0.80	3.08%	0.5	0.13	0.10	24.36%
	SVM	0.5	0.65	0.67	3.87%	0.5	0.10	0.10	1.62%
	Neural	0.1	0.64	0.74	14.47%	0.5	0.56	0.10	81.91%
	NB	0.5	0.66	0.70	6.16%	0.5	0.30	0.17	42.85%
	C5.0	0.5	0.50	0.65	31.70%	1.0	0.10	0.16	-52.02%
	PLS	1.0	0.58	0.60	4.08%	0.5	0.35	0.34	0.02%
49.69	10NN	0.1	0.44	0.77	75.49%	0.1	197.65	73.12	63.00%
	LDA	0.1	0.82	0.82	0.10%	0.5	0.10	0.09	7.50%
	Logistic	0.1	0.78	0.79	0.76%	0.1	0.09	0.09	1.66%
	SVM	0.1	0.67	0.71	5.86%	0.1	0.10	0.09	1.39%
	Neural	0.1	0.76	0.79	5.18%	1.0	0.20	0.11	43.14%
	NB	0.1	0.57	0.58	0.53%	0.1	0.13	0.13	2.43%

Continued on next page

Table 4.2 – continued from previous page

IR	Model	Opt. σ_{noise}	ROC/AUC Orig.	ROC/AUC Noisy	% inc.	Opt. σ_{noise}	KL distance Orig.	KL distance Noisy	% dec.
	C5.0	0.5	0.50	0.56	12.95%	1.0	0.10	0.30	-210.47%
	PLS	0.1	0.83	0.83	0.01%	1.0	0.34	0.34	0.09%
58.28	10NN	0.5	0.44	0.69	55.41%	0.1	202.69	14.03	93.08%
	LDA	1.0	0.69	0.80	15.90%	1.0	0.11	0.08	21.92%
	Logistic	1.0	0.71	0.77	7.27%	0.5	0.15	0.08	47.17%
	SVM	0.5	0.79	0.81	2.93%	0.1	0.08	0.07	7.15%
	Neural	0.5	0.56	0.75	35.05%	0.5	0.28	0.08	71.63%
	NB	1.0	0.77	0.80	4.08%	0.5	0.32	0.21	34.10%
	C5.0	1.0	0.51	0.78	53.72%	0.5	0.09	0.21	-132.63%
	PLS	1.0	0.56	0.58	1.88%	1.0	0.34	0.34	0.04%
58.40	10NN	0.5	0.40	0.71	78.54%	0.5	203.38	29.24	85.62%
	LDA	1.0	0.57	0.60	6.01%	1.0	0.10	0.09	8.18%
	Logistic	1.0	0.56	0.58	2.52%	1.0	0.10	0.10	4.87%
	SVM	0.1	0.93	0.90	-3.22%	0.1	0.04	0.05	-6.83%
	Neural	0.1	0.58	0.69	18.82%	0.5	0.42	0.08	80.16%
	NB	0.1	0.57	0.57	0.20%	0.5	0.10	0.10	3.51%
	C5.0	0.1	0.54	0.82	53.18%	1.0	0.09	0.11	-29.66%
	PLS	0.5	0.55	0.55	0.99%	1.0	0.34	0.34	0.02%
66.67	10NN	1.0	0.66	0.99	51.17%	1.0	225.75	225.02	0.33%
	LDA	1.0	1.00	1.00	0.02%	1.0	0.48	0.28	41.77%
	Logistic	1.0	1.00	1.00	0.40%	1.0	0.17	0.00	98.50%
	SVM	0.1	1.00	1.00	0.00%	0.1	0.00	0.00	-18.23%
	Neural	0.5	0.97	1.00	3.43%	0.5	0.02	0.01	55.84%
	NB	0.5	0.99	0.99	0.37%	0.1	1.56	1.62	-3.94%
	C5.0	0.1	1.00	1.00	0.36%	0.1	0.00	0.00	49.35%
	PLS	0.5	0.97	0.97	0.39%	0.5	0.33	0.33	0.10%
72.69	10NN	0.1	0.70	0.95	34.91%	0.1	219.05	154.61	29.42%
	LDA	0.1	0.97	0.97	-0.19%	1.0	0.08	0.04	48.64%
	Logistic	0.1	0.97	0.97	0.17%	0.1	0.10	0.04	61.84%
	SVM	0.1	0.85	0.87	2.95%	0.1	0.06	0.05	7.65%
	Neural	0.1	0.90	0.95	5.94%	0.1	0.17	0.04	77.08%
	NB	0.1	0.80	0.80	0.18%	1.0	0.46	0.27	41.15%
	C5.0	0.5	0.64	0.85	33.53%	1.0	0.07	0.09	-29.05%
	PLS	0.1	0.97	0.97	0.00%	0.5	0.33	0.33	0.01%
85.88	10NN	0.5	0.45	0.61	36.71%	0.5	205.74	37.54	81.76%
	LDA	0.5	0.60	0.60	-0.18%	1.0	0.08	0.07	7.87%
	Logistic	1.0	0.60	0.61	1.04%	0.1	0.08	0.08	4.12%
	SVM	0.1	0.94	0.91	-3.50%	0.1	0.04	0.04	-7.65%
	Neural	0.1	0.56	0.62	11.32%	0.1	0.47	0.06	86.69%
	NB	1.0	0.57	0.62	9.90%	1.0	0.09	0.08	6.37%

Continued on next page

Table 4.2 – continued from previous page

IR	Model	Opt. σ_{noise}	ROC/AUC Orig.	ROC/AUC Noisy	% inc.	Opt. σ_{noise}	KL distance Orig.	KL distance Noisy	% dec.
	C5.0	1.0	0.55	0.70	26.20%	1.0	0.06	0.16	-160.22%
	PLS	1.0	0.58	0.59	1.05%	1.0	0.33	0.33	0.01%
129.44	10NN	0.1	0.41	0.66	61.08%	0.1	214.69	176.41	17.83%
	LDA	0.1	0.85	0.85	0.17%	0.5	0.06	0.05	14.79%
	Logistic	0.1	0.80	0.80	0.42%	0.5	0.05	0.04	20.56%
	SVM	0.1	0.63	0.70	10.41%	0.1	0.04	0.04	0.72%
	Neural	0.1	0.80	0.84	5.02%	0.1	0.09	0.04	56.41%
	NB	1.0	0.69	0.71	2.26%	1.0	0.27	0.19	29.14%
	C5.0	1.0	0.50	0.69	37.66%	1.0	0.05	0.24	-433.21%
	PLS	1.0	0.72	0.77	6.70%	1.0	0.32	0.32	0.01%

Chapter 5

Noisy Replication for Imbalanced Multi-Class Data Sets

The previous two chapters have proved that the noisy replication is an effective model-free method in learning the imbalanced binary data, and this chapter will expand this research on the imbalanced multi-class data set. Multi-class means there are at least three categories of the response variable in the data set. The majority class of a multi-class data set is the class whose number of observations is the largest compared with other classes; the minority class is the class with the least number of observations in that class. We can randomly pick one class as the majority class if there are two or more classes having the same largest amount of observations, and the same rule applied for selecting the minority class. Same as in the binary data set, the imbalance ratio of a multi-class data set is defined as the number of observations in the majority class over that of the minority class. Data sets listed in Table 4.1 with *Classes* equal to and larger than 3 will be tested in this Chapter. The imbalance ratio of these data sets has a huge leap, ranging from 1.5 to 853.

In this chapter, we will first adjust the noisy replication method to fit the multi-class data set. When applying the noisy replication method to the multi-class data set, After testing with both the simulated and real data sets, we display their outcomes, and compare the effectiveness of different noise levels, models, and imbalance ratios.

5.1 Method Adjustment

The approach of adding noisy replicates to the multi-class data set is generally similar to the binary data set: select the minority class, add the noise (σ_{noise} = 0.1, 0.5, and 1.0), apply the 2-fold cross-validation is applied, test eight models, and run the simulation for 100 times.

Two changes are designed specifically to learn the multi-class data set. As for the

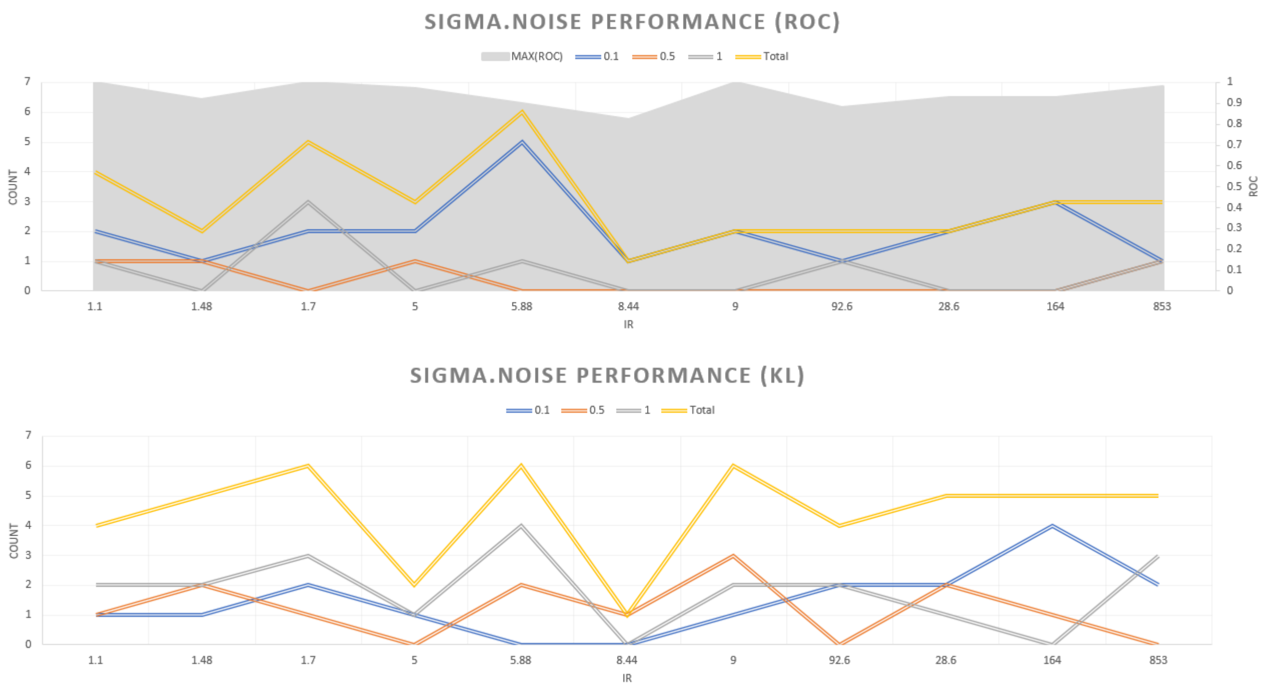


Figure 5.1: Counts of optimal models in each multi-class data set

measurement, the AUC value is defined by the average of AUC values of all possible pairs of two classes. For instance, in a three-class data set, there are $\binom{3}{2}$ pairs of combinations, and an AUC value could be calculated for each pair. Hence, the AUC value for this three-class data set is the mean of three AUC values [20]. This method was further explained in Chapter 2. As for some models, we also update their R functions in learning the multi-class data set. For instance, `kknn` is for KNN and `multinom` is for logistic regression [12, 23].

5.2 Results and Interpretations

Table 5.1 at the end of this chapter summarizes the optimal results for the noisy replication method, and Appendix C plots the 95% confident interval of $\triangle$ROC and $\triangle$KL-distance for each data set. Several figures are plotted to help us interpret Table 5.1 and Appendix C.

Two plots in Figure 5.1 summarize the total number of optimal models for each

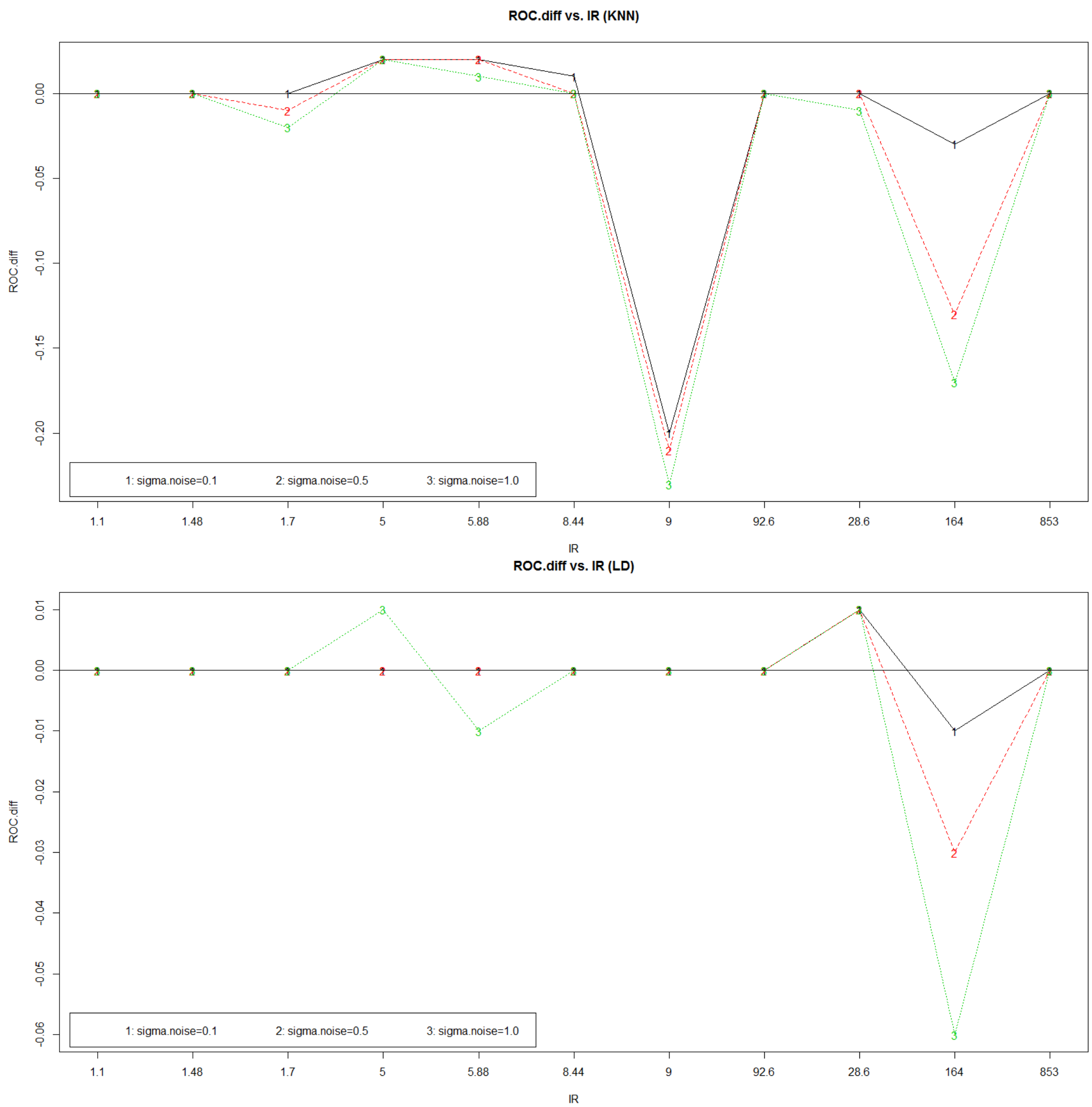
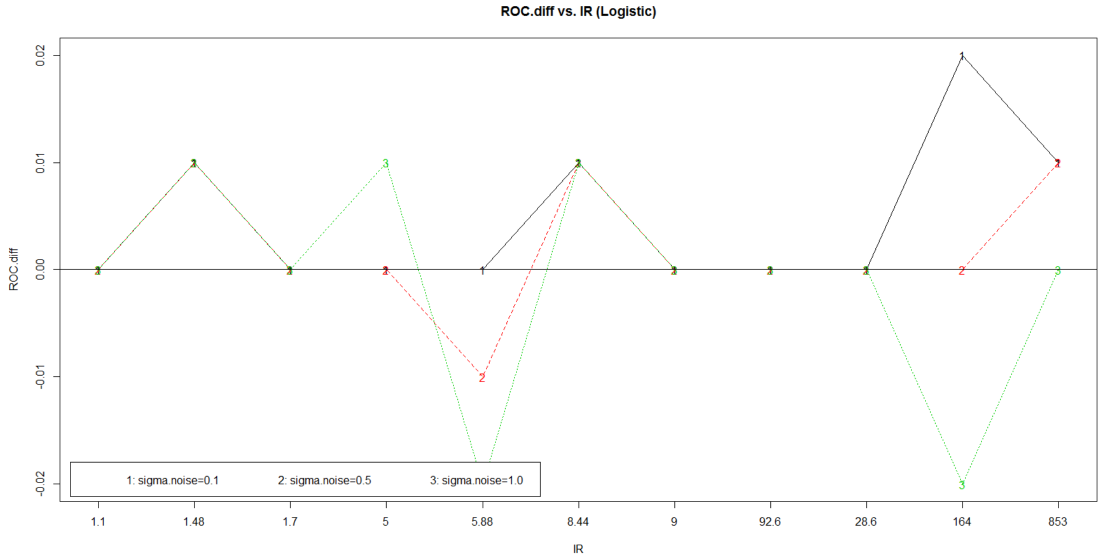


Figure 5.2: $\triangle$ROC vs. IRs in eight models for multi-class data sets

data set. The blue plot represents the performance when $\sigma_{noise} = 0.1$, the red plot represents $\sigma_{noise} = 0.5$, and the gray plot represents $\sigma_{noise} = 1.0$. The yellow plot represents the total number of optimal models for all data sets, and its number should be less than or equal to eight (8), the number of machine learning models. The gray shaded area illustrates the best AUC values a data set could get with noisy replicates. The top figure is measured by ROC, while the buttom one is by KL distance. Assessing with the ROC area, models with $\sigma_{noise} = 0.1$ perform better than other noise levels, while models with $\sigma_{noise} = 0.5$ do not perform very well especially



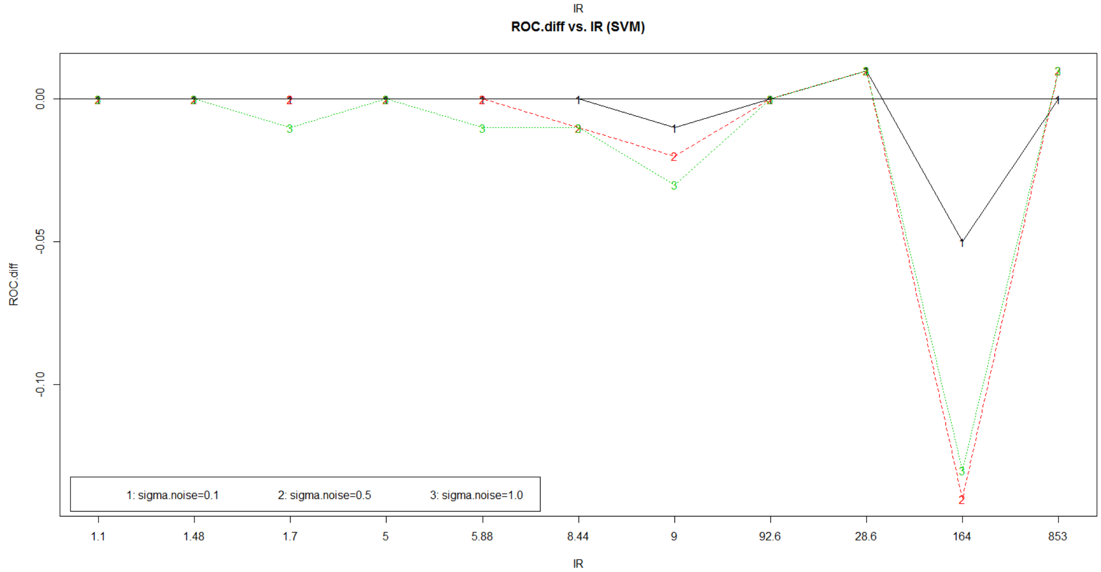


Figure 5.2: $\triangle$ROC vs. IRs in eight models for multi-class data sets (cont.)

when IR is relatively large. In addition, most optimal models have an AUC value greater than 0.85. Assessing with the KL distance, models with $\sigma_{noise} = 0.1$ perform better when IR is relatively large, while models with $\sigma_{noise} = 1.0$ perform better when IR is relatively small. For both assessments, could we bring more data sets in, a better plot about the model performance would be generated.

A series of plots in Figure 5.2 illustrate the relationship between $\triangle$ROC (`ROC.diff`) and IR for eight commonly used machine learning models. From these plots, we can tell that models, such as neural network and C5.0, will have a better performance

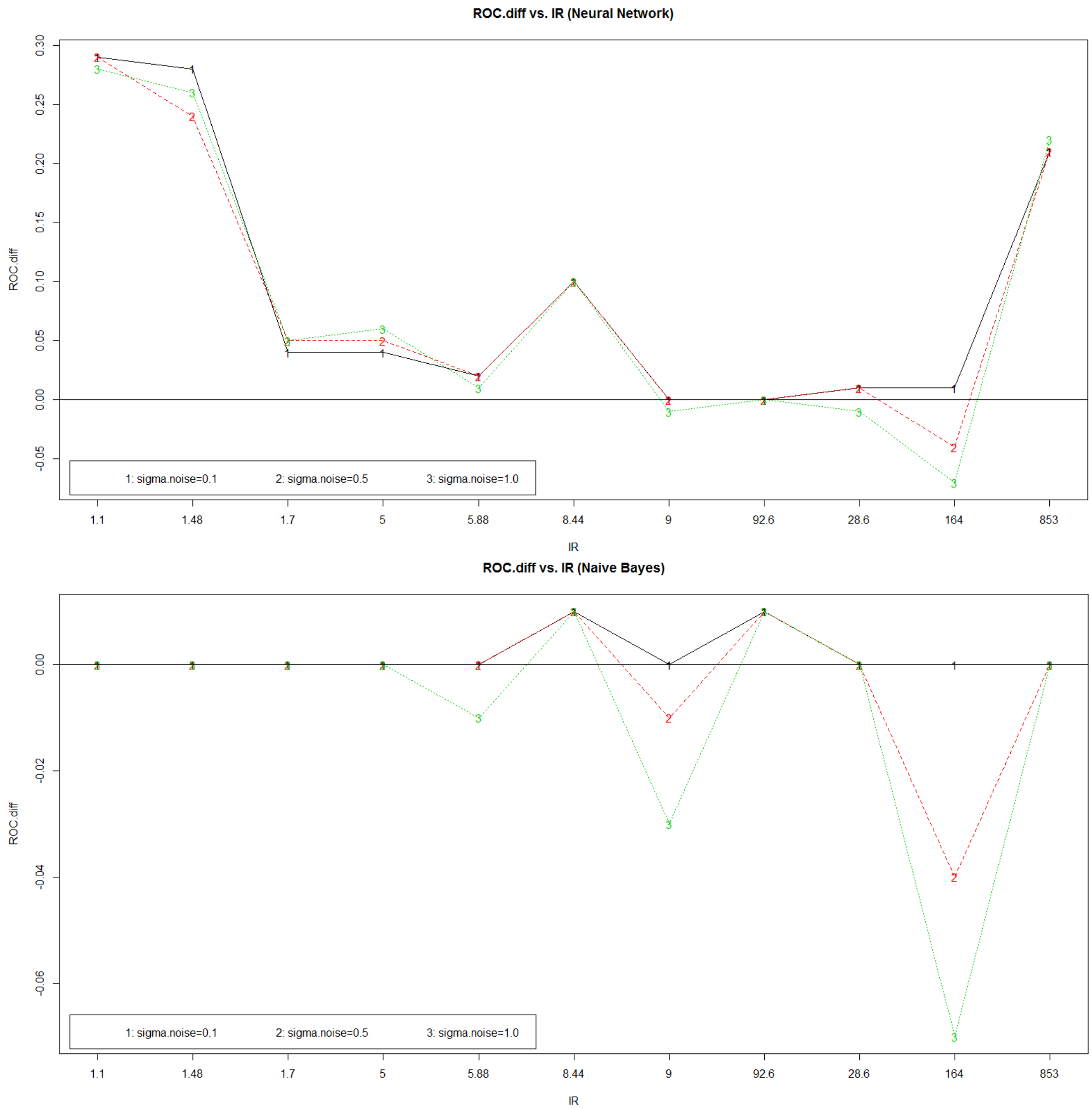


Figure 5.2: △ROC vs. IRs in eight models for multi-class data sets (cont.)

especially when the imbalance ratio of a data set is relatively small. However, we can also observe some "failures" as well. Data sets with relatively large IR, such as IR = 164, have an unstable performance varying among models. In addition, many other models, Logistic Regression, SVM, and Naïve Bayes, for instance, do not receive a statistically significant increase of △ROC with the change of IR. Nevertheless, this does not mean the noisy replication method is not successful in learning the imbalanced multi-class data set. This phenomenon is because AUC values of both the original models and the optimized models are very close to 1.0, the best AUC value

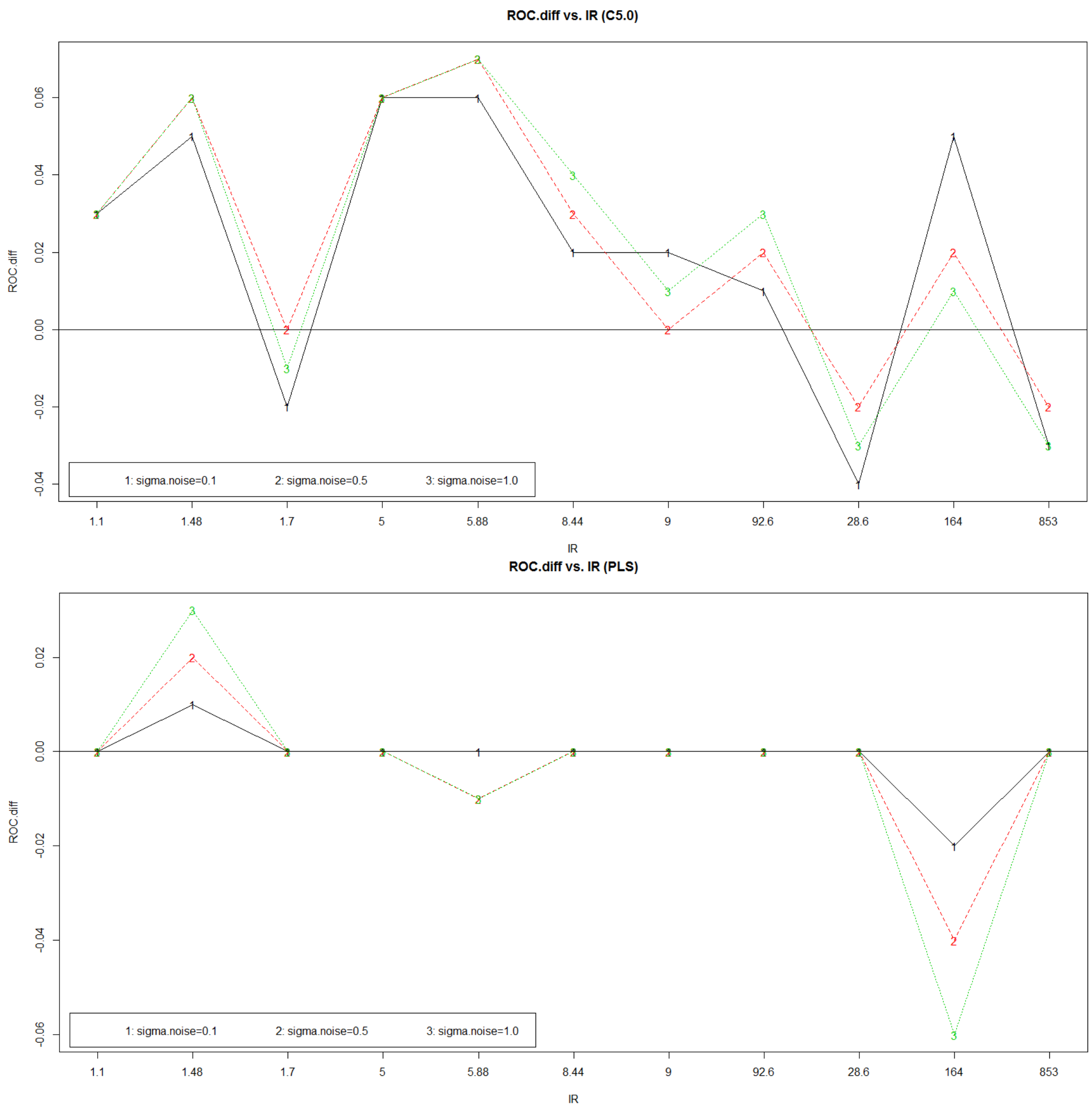


Figure 5.2: △ROC vs. IRs in eight models for multi-class data sets (cont.)

could be made. Consequently, there is little space to improve the prediction (Figure 5.1). If more multi-class real data sets are available, a clearer tendency between the ROC difference and IR could be observed.

Figure 5.3 demonstrates the relationship between △ROC (`ROC.diff`) and IR for three noise levels (σ_{noise}). When $\sigma_{noise} = 0.1$, there is an increase of △ROC for most machine learning models after adding noise replicates, and among them the neural network has an outstanding performance. We can also observe that the noisy replication method has a better and more stable performance when IR is less than 9.

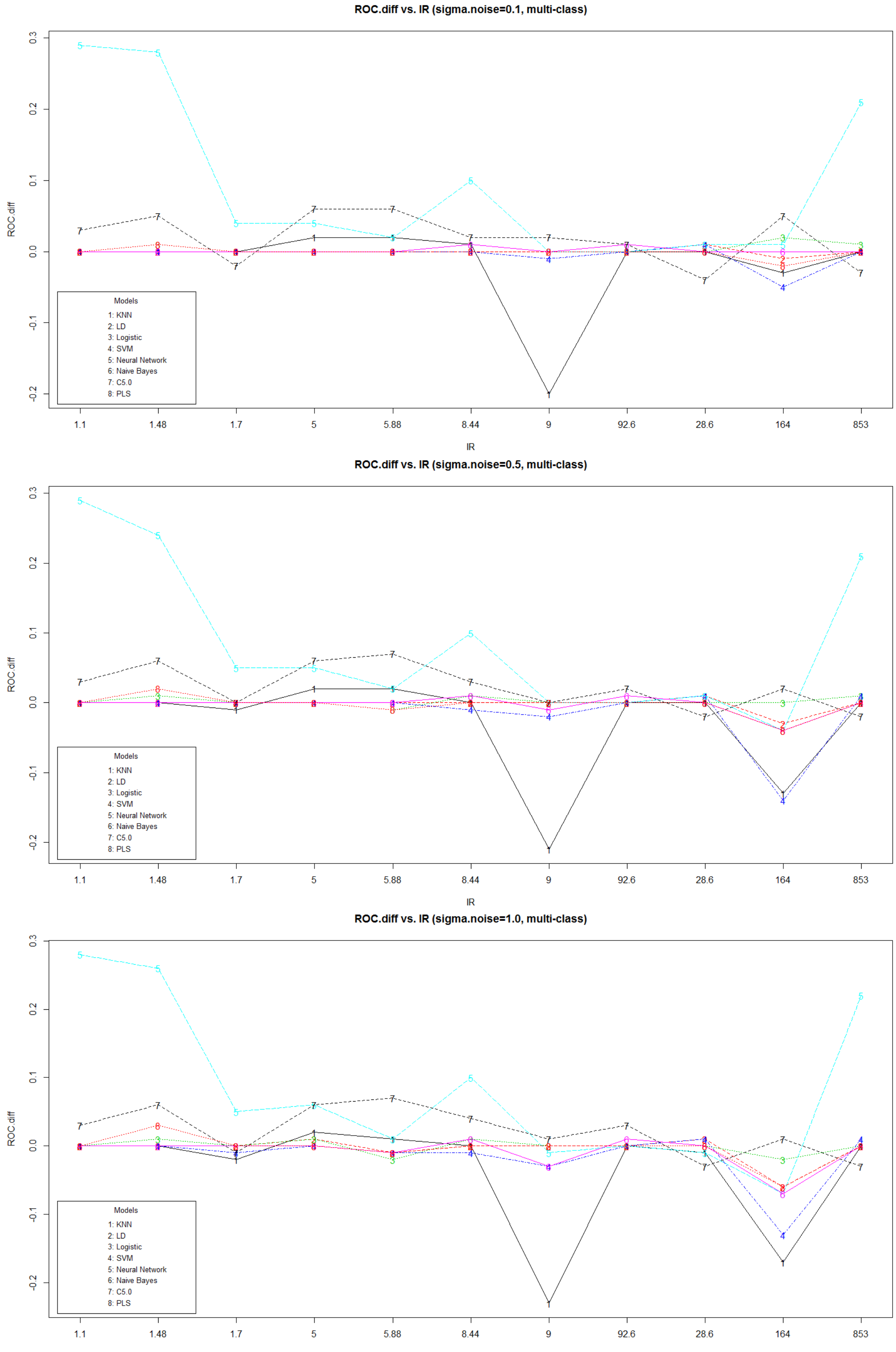


Figure 5.3: Model performance with different noise levels in multi-class data sets

However, when IR = 164, $\triangle$ROC drops for many models, such as SVM and KNN. The same phenomena also happen to $\sigma_{noise} = 0.5$ and $\sigma_{noise} = 1.0$. The limited number of real data sets and the sparseness of the IR could be the factors.

Table 5.1: Optimal noise level for each multi-class data set

IR	Model	Opt. σ_{noise}	ROC/AUC Orig.	ROC/AUC Noisy	% inc.	Opt. σ_{noise}	KL distance Orig.	KL distance Noisy	% dec.
1.10	10NN	0.1	0.99	0.99	0.00%	0.5	2.77	2.03	26.71%
	LDA	0.1	0.99	0.99	0.00%	0.5	0.59	0.55	6.78%
	Logistic	0.1	0.99	0.99	0.00%	1.0	0.46	0.37	19.57%
	SVM	0.1	1.00	1.00	0.00%	1.0	1.06	1.05	0.94%
	Neural	0.1	0.68	0.97	42.65%	0.1	1.97	1.74	11.68%
	NB	0.1	0.98	0.98	0.00%	0.5	1.88	1.86	1.06%
	C5.0	0.1	0.96	0.98	2.08%	1.0	0.68	1.60	-135.29%
	PLS	0.1	0.85	0.85	0.00%	0.1	2.17	2.17	0.00%
1.48	10NN	0.1	1.00	1.00	0.00%	1.0	0.29	0.12	58.62%
	LDA	0.1	1.00	1.00	0.00%	0.5	0.08	0.05	37.50%
	Logistic	0.1	0.98	0.99	1.02%	0.1	2.92	0.43	85.27%
	SVM	0.1	1.00	1.00	0.00%	0.1	0.11	0.11	0.00%
	Neural	0.1	0.71	1.00	40.85%	1.0	0.96	0.51	46.88%
	NB	0.1	1.00	1.00	0.00%	0.1	0.12	0.13	-8.33%
	C5.1	0.5	0.93	0.99	6.45%	1.0	0.41	0.55	-34.15%
	PLS	1.0	0.90	0.93	3.33%	0.1	0.85	0.85	0.00%
1.70	10NN	0.1	0.76	0.76	0.00%	0.5	1.61	0.93	42.24%
	LDA	0.1	0.83	0.82	-1.20%	1.0	0.83	0.78	6.02%
	Logistic	0.1	0.81	0.82	1.23%	0.1	1.07	0.84	21.50%
	SVM	0.1	0.87	0.87	0.00%	0.1	0.65	0.67	-3.08%
	Neural	0.5	0.83	0.89	7.23%	0.5	1.18	0.66	44.07%
	NB	0.1	0.87	0.87	0.00%	1.0	0.69	0.66	4.35%
	C5.2	0.5	0.92	0.92	0.00%	0.5	0.48	1.15	-139.58%
	PLS	0.1	0.78	0.78	0.00%	0.1	0.96	0.96	0.00%
5.00	10NN	0.1	0.97	0.99	2.06%	0.5	3.67	0.44	88.01%
	LDA	1.0	0.99	1.00	1.01%	1.0	0.31	0.24	22.58%
	Logistic	1.0	0.99	1.00	1.01%	1.0	1.27	0.08	93.70%
	SVM	0.1	0.99	0.99	0.00%	0.1	0.15	0.14	6.67%
	Neural	1.0	0.94	1.00	6.38%	1.0	0.33	0.17	48.48%
	NB	0.1	1.00	1.00	0.00%	0.1	0.18	0.17	5.56%
	C5.3	0.1	0.91	0.97	6.59%	1.0	0.35	2.78	-694.29%
	PLS	0.1	0.89	0.89	0.00%	0.1	0.80	0.80	0.00%
5.88	10NN	0.1	0.85	0.88	3.53%	1.0	3.76	0.40	89.36%
	LDA	0.1	0.94	0.94	0.00%	0.1	0.33	0.34	-3.03%
	Logistic	0.1	0.96	0.96	0.00%	0.1	0.28	0.28	0.00%
	SVM	0.1	0.94	0.94	0.00%	0.1	0.29	0.29	0.00%
	Neural	0.1	0.95	0.97	2.11%	0.1	0.34	0.25	26.47%
	NB	0.1	0.87	0.87	0.00%	0.1	0.49	0.49	0.00%
	C5.4	0.5	0.74	0.81	9.46%	0.1	0.70	6.17	-781.43%

Continued on next page

Table 5.1 – continued from previous page

IR	Model	Opt. σ_{noise}	ROC/AUC Orig.	ROC/AUC Noisy	% inc.	Opt. σ_{noise}	KL distance Orig.	KL distance Noisy	% dec.
	PLS	0.1	0.91	0.90	-1.10%	0.1	0.76	0.77	-1.32%
8.44	10NN	0.1	0.86	0.87	1.16%	1.0	13.68	10.71	21.71%
	LDA	0.1	0.86	0.87	1.16%	0.5	1.86	1.67	10.22%
	Logistic	0.1	0.85	0.86	1.18%	0.5	17.04	7.38	56.69%
	SVM	0.1	0.90	0.90	0.00%	1.0	1.23	1.19	3.25%
	Neural	0.1	0.78	0.88	12.82%	1.0	1.70	0.99	41.76%
	NB	0.1	0.82	0.83	1.22%	1.0	6.68	5.59	16.32%
	C5.5	1.0	0.82	0.86	4.88%	1.0	1.18	20.06	-1600.00%
	PLS	0.1	0.79	0.79	0.00%	0.1	1.58	1.58	0.00%
9.00	10NN	0.1	1.00	0.80	-20.00%	1.0	0.02	0.57	-2750.00%
	LDA	0.1	0.82	0.82	0.00%	0.1	0.38	0.38	0.00%
	Logistic	0.1	0.82	0.82	0.00%	0.1	0.38	0.38	0.00%
	SVM	0.1	0.72	0.71	-1.39%	0.1	0.41	0.41	0.00%
	Neural	0.1	0.81	0.81	0.00%	0.5	0.40	0.38	5.00%
	NB	0.1	0.81	0.81	0.00%	0.1	0.39	0.39	0.00%
	C5.6	0.1	0.72	0.75	4.17%	1.0	0.5	1.57	-214.00%
	PLS	0.1	0.82	0.82	0.00%	0.1	0.74	0.74	0.00%
92.60	10NN	0.1	0.84	0.84	0.00%	1.0	17.18	16.89	1.69%
	LDA	0.1	0.88	0.88	0.00%	1.0	2.07	2.03	1.93%
	Logistic	0.1	0.88	0.88	0.00%	0.1	1.13	1.13	0.00%
	SVM	0.1	0.84	0.84	0.00%	1.0	2.60	2.60	0.00%
	Neural	0.1	0.83	0.83	0.00%	0.1	1.21	1.17	3.31%
	NB	0.1	0.83	0.85	2.41%	0.1	3.29	3.11	5.47%
	C5.7	1.0	0.77	0.79	2.60%	1.0	1.51	28.14	-1763.58%
	PLS	0.1	0.73	0.73	0.00%	0.1	2.10	2.10	0.00%
28.60	10NN	0.1	0.93	0.93	0.00%	0.5	7.52	7.35	2.26%
	LDA	0.1	0.81	0.81	0.00%	0.5	22.48	16.22	27.85%
	Logistic	0.1	0.93	0.93	0.00%	0.1	1.48	1.27	14.19%
	SVM	0.1	0.86	0.87	1.16%	1.0	1.58	1.56	1.27%
	Neural	0.1	0.87	0.88	1.15%	0.1	0.88	0.61	30.68%
	NB	0.1	0.93	0.93	0.00%	0.5	1.98	1.98	0.00%
	C5.8	0.5	0.84	0.82	-2.38%	0.5	0.82	22.22	-2609.76%
	PLS	0.1	0.74	0.74	0.00%	0.1	1.65	1.65	0.00%
164.00	10NN	0.1	0.91	0.88	-3.30%	0.1	3.05	3.44	-12.79%
	LDA	0.1	0.95	0.93	-2.11%	0.5	1.02	0.33	67.65%
	Logistic	0.1	0.89	0.91	2.25%	0.1	4.73	0.56	88.16%
	SVM	0.1	0.86	0.81	-5.81%	0.1	0.37	0.35	5.41%
	Neural	0.1	0.80	0.81	1.25%	0.1	0.38	0.28	26.32%
	NB	0.1	0.92	0.92	0.00%	0.1	3.40	2.20	35.29%
	C5.9	0.1	0.87	0.93	6.90%	0.1	0.24	2.07	-762.50%

Continued on next page

Table 5.1 – continued from previous page

IR	**Model**	**Opt.** σ_{noise}	**ROC/AUC Orig.**	**ROC/AUC Noisy**	**% inc.**	**Opt.** σ_{noise}	**KL distance Orig.**	**KL distance Noisy**	**% dec.**
	PLS	0.1	0.62	0.60	-3.23%	0.1	1.06	1.06	0.00%
853.00	10NN	0.1	0.94	0.94	0.00%	0.1	0.38	0.38	0.00%
	LDA	0.1	0.95	0.95	0.00%	1.0	0.44	0.38	13.64%
	Logistic	0.1	0.96	0.97	1.04%	0.1	0.60	0.33	45.00%
	SVM	0.5	0.97	0.98	1.03%	1.0	1.19	1.17	1.68%
	Neural	1.0	0.75	0.97	29.33%	1.0	0.33	0.19	42.42%
	NB	0.1	0.96	0.96	0.00%	0.1	0.84	0.77	8.33%
	C5.10	0.5	0.90	0.88	-2.22%	0.5	0.03	0.86	-2766.67%
	PLS	0.1	0.68	0.68	0.00%	0.1	1.11	1.11	0.00%

Chapter 6

Conclusion

This thesis mainly proves the effectiveness of the noisy replication method in learning either the imbalanced binary data set or the imbalanced multi-class data set. By applying the noisy replication method to more than 60 simulated and real data sets, we gained further understanding of this machine learning approach, which is a mixture of many components. To achieve a higher AUC value or a smaller KL distance, we need many "tuning" many factors, such as the model selected, the noise level added, the imbalance ratio of the data set, the number of classes, the data type (quantitative or qualitative), noise vibration direction, assessment criteria, etc. This thesis also provides us clear clues to answer questions from the first chapter.

What kind of noise should be added? Three levels of noise are tested in this thesis. For each individual data set, we examine the performance of each noise level by comparing the ROC area and the KL distance between eight commonly used machine learning models and their improved models with noisy replicates. It is certain that adding noise could improve the prediction outcome, and by increasing the noise, the noisy replication model could either perform better, worse, or with even no difference. However, we still could see that $\sigma_{noise} = 0.5$ performs better and more stable especially when the imbalance ratio increases, and more noisy replication models will generate a positive increase of the ROC area and a negative increase of the KL distance.

Where should the noise be added, majority class, minority class, or both? The tradeoff between variance and bias determines how the noisy replication method is applied. Controlling other factors, such as noise level, repeated times, etc., adding noise to either the majority class or the minority class will lower the variance; however, adding noise only to the majority class may further reduce the bias than only adding noise to the minority class. We also expect to make the minimum change to the original data set. Therefore, adding noise only to the minority class is a better idea.

As for the multi-class data set, where the number of observations in several classes are either small or large, adding noise only to the minority class, the class with the least number of observations, is proved to be effective in Chapter 5. Nevertheless, the effectiveness of adding noise to several classes with smaller number of observations is still waiting to be tested in future research.

How many times should the minority classes be repeated? There is one factor, noisy.train, in the pseudocode controlling the repeat times of the minority class. In the prototype study with a simulated imbalanced data set, we did not observe a statistically significant increase in the model performance. Hence, noisy.train is defined as 1 for all other real data set tests. However, this does not mean the repeat of the minority class is useless, and we need further investigation on this topic in future research.

Will anti-noise, or two-side vibration improve the performance? This has been discussed in Chapter 3 where the noisy replication algorithm and its pseudocode are introduced. Traditionally, only one noise will be added to the minority class, which is defined as one-side vibration. It has a decent performance if the minority class adds the noise and minuses the noise (anti-noise) in a simulated binary data set in Chapter 3. We name it as two-side vibration, and adopt it to all other real data sets. To conclude, adding both noise and anti-noise is a successful trial in the noisy replication method.

Can this algorithm be applied to both the qualitative data and the quantitative data? Most data sets in this thesis consist only of quantitative data, i.e., all features are real numbers, and the noisy replication method is an effective learning approach. As for the qualitative data, nevertheless, the noisy replication method could not deal with the nominal data or the integer data, since it does not make sense to “vibrate” male or female, or yellow to red. Therefore, a mixture of the noisy replication method and other techniques need to be applied. It is recommended to add noise only to the quantitative data and leave the qualitative data as it is, assuming the original machine learning model could deal with both qualitative data and quantitative data.

Will this method be applied to the data set with multiple classes? The answer is yes. Chapter 5 provides a prototype in testing the effectiveness of the noisy replication method in imbalanced multi-class data sets. Many models demonstrate a statistically significant increase in ROC or a decrease in KL distance by adding a proper noise level in the minority class. More data sets could be tested in future research.

Will the imbalance ratio (IR) influence on the model performance? There is a positive correlation between the $\triangle$ROC and IR for some models, such as KNN and neural network, in the binary data set. On the one hand, it further proves the effectiveness of the noisy replication method; on the other hand, it is not responsible to conclude a causal relationship between them. As for the multi-class data set, due to the limited number of real data sets, it is hard to see a strong relationship between $\triangle$ROC and IR.

How to measure and assess the performance of the algorithm, such as ROC area and Kullback-Leibler distance? Which one is better? The performance of these two assessment criteria is not consistent, which means an increasing (decreasing) AUC value does not have a decreasing (increasing) KL distance counterpart. As for the multi-class data set, multiple ROC is adopted. It is hard to conclude which one is better than the other, and that's the reason why we keep both measurement results in this thesis.

Which model performs better with the noisy replicates? In this thesis, there are eight (8) machine learning models selected in testing the performance of the noisy replication method: KNN, LDA, Logistic, SVM, Neural Network, Naïve Bayes, C5.0, and PLS. Generally speaking, KNN, Neural Network, Naïve Bayes, and PLS have a better performance in terms of both the ROC area and the KL-distance measurement. However, the performance of LD, Logistic Regression, and SVM do not always generate a desirable outcome either in the ROC area or the KL distance. This is also applied to the multi-class data sets.

How good is the noisy replication method compared with other algorithms using the same real data set? There are many other previous researches adopting the same KEEL data sets as we did, providing us a good test for our results. As a result, we could see that some of our results are better than theirs, while some are not. This is because some papers focus on improving a certain method, such as SVM, decision tree, etc. [24, 3]; some redesign the assessment method according to their need; some only focus on binary data sets; some only apply their method to a limited number of data sets. In general, the performance of the noisy replication method demonstrates that it is a model-free regularization method in learning the imbalanced data set.

This thesis makes a great leap in learning, predicting, and classifying the imbalanced data set with the noisy replication method. Meanwhile, there are several topics we are interested in the future research: Non-static imbalanced data set. What happens if the minority class changes in the real-time? More multi-class data sets. What is the relationship between IR and model performance? Noisy replication method in a more balanced data set. Can we apply the same methodology for other general data sets, even though they are not imbalanced? This thesis provides some samples in dealing with data sets with a lower IR, and future researches could study this topic further.

References

[1] G. James, D. Witten, and T. Hastie, "An introduction to statistical learning: With applications in r.," 2014.

[2] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[3] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, and Y. Zhou, "A novel ensemble method for classifying imbalanced data," *Pattern Recognition*, vol. 48, no. 5, pp. 1623–1637, 2015.

[4] S. S. Lee, "Noisy replication in skewed binary classification," *Computational statistics & data analysis*, vol. 34, no. 2, pp. 165–191, 2000.

[5] H. He and E. A. Garcia, "Learning from imbalanced data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 9, pp. 1263–1284, 2009.

[6] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, pp. 113 – 141, 2013.

[7] L. A. Jeni, J. F. Cohn, and F. De La Torre, "Facing imbalanced data–recommendations for the use of performance metrics," in *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*, pp. 245–251, IEEE, 2013.

[8] M. Kuhn and K. Johnson, *Applied predictive modeling*, vol. 26. Springer, 2013.

[9] Q. Yang, X. Wu, P. Domingos, C. Elkan, J. Gehrke, J. Han, D. Heckerman, D. Keim, J. Liu, D. Madigan, G. Piatetsky-Shapiro, V. V. Raghavan, R. Rastogi, S. J. Stolfo, A. Tuzhilin, and B. W. Wah, "10 Challenging Problems in Data

Mining Research," *International Journal of Information Technology & Decision Making*, vol. 5, no. 4, pp. 597–604, 2006.

[10] A. Fernández, S. García, M. J. del Jesus, and F. Herrera, "A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets," *Fuzzy Sets and Systems*, vol. 159, no. 18, pp. 2378–2398, 2008.

[11] S. S. Lee, "Regularization in skewed binary classification," *Computational Statistics*, vol. 14, no. 2, p. 277, 1999.

[12] K. Schliep and K. Hechenbichler, *kknn: Weighted k-Nearest Neighbors*, 2016. R package version 1.3.1.

[13] W. N. Venables and B. D. Ripley, *Modern applied statistics with S-PLUS*. Springer Science & Business Media, 2013.

[14] Y. Croissant *et al.*, "Estimation of multinomial logit models in r: The mlogit packages," *R package version 0.2-2. URL: http://cran. r-project. org/web/packages/mlogit/vignettes/mlogit. pdf*, 2012.

[15] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*. New York: Springer, fourth ed., 2002. ISBN 0-387-95457-0.

[16] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch, *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2017. R package version 1.6-8.

[17] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*. New York: Springer, fourth ed., 2002. ISBN 0-387-95457-0.

[18] M. Kuhn, S. Weston, N. Coulter, and M. C. C. code for C5.0 by R. Quinlan, *C50: C5.0 Decision Trees and Rule-Based Models*, 2015. R package version 0.1.0-24.

[19] M. K. C. from Jed Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, the R Core Team, M. Benesty, R. Lescarbeau, A. Ziem, L. Scrucca, Y. Tang, C. Candan, and T. Hunt., *caret: Classification and Regression Training*, 2016. R package version 6.0-73.

[20] D. J. Hand and R. J. Till, "A simple generalisation of the area under the roc curve for multiple class classification problems," *Machine learning*, vol. 45, no. 2, pp. 171–186, 2001.

[21] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[22] M. Lichman, "UCI machine learning repository," 2013.

[23] X. Robin, N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.-C. Sanchez, and M. Müller, "proc: an open-source package for r and s+ to analyze and compare roc curves," *BMC bioinformatics*, vol. 12, no. 1, p. 77, 2011.

[24] J. F. Díez-Pastor, J. J. Rodríguez, C. García-Osorio, and L. I. Kuncheva, "Random balance: ensembles of variable priors classifiers for imbalanced data," *Knowledge-Based Systems*, vol. 85, pp. 96–111, 2015.

Appendices

A Sample Code

This is the sample R code based on the pilot experiment introduced in Chapter 3.

```
# import libraries
library(MASS)
library(class)
library(ROCR)
library(kknn)    # knn
library(e1071)   # SVM
library(rpart)   # Tree
library(tree)
library(klaR)    # NB
library(C50)
library(pls)
library(caret)   # plsda
library(mda)
library(nnet)    # multinom
library(Rmisc)   # plot
library(ggplot2)
library(mvtnorm)
library(pROC)
library(verification)
library(randomForest)
library(matrixStats)

par(mfrow=c(3,1))
ptm<-proc.time()

# c("knn","ld","log","svm","dtree","ptree","neural","nb","C50","fda","pls","mda")

for (models in c("dtree","ptree","neural","nb","C50","fda","pls","mda")) {
  print (models)

  N<-2                                      # number of varialbes (x)
  sigma.noise<-c(0.1,0.5,1.0)               # test for 0, 0.1, 0.5, 1.0
  nsim<-100                                 # 3, 100, simulation repeat times (1, 50, 100, 500)
  noisy.repl<-c(1)                          # c(1:3), c(1:10) replications of the rare parts (y=1)
  noisy.train<-c(10,20,40,60,80,100)        # c(1:10), c(10,20,40,60,80,100) replications of the training data set
  nnrepl<-max(noisy.repl)                   # number of rows; maximum number in the noisy.repl; j
  nntrain<-max(noisy.train)                 # number of columns; maximum number in the noisy.train; i
```

```
n0<-200                                    # number of "0"s in the training data set
n1<-20                                     # number of "1"s in the training data set
n2<-180
train.size<-220                            # size of the training data set
e<-0.00000001                              # for KL distance
roc.diff.ci<-c()                           # for the plot
roc.diff.mean<-c()                         # for the plot
kl.diff.ci<-c()
kl.diff.mean<-c()
eu.diff.ci<-c()
eu.diff.mean<-c()
kValue<-10                                 # k for knn (1,10)

for (k in sigma.noise) {
  cat("\nmodel =", models, "; sigma.noise =", k, "\n")

  # evaluation criteria
  roc.multi<-list()
  rocdiff.multi<-list()
  roc.sum<-matrix(0, nrow=nnrepl, ncol=nntrain) # save roc results for each nsim; same as roc.ave in previous versions
  rocdiff.sum<-matrix(0, nrow=nnrepl, ncol=nntrain)
  rocMean<-0

  kl.multi<-list()
  kldiff.multi<-list()
  kl.sum<-matrix(0, nrow=nnrepl, ncol=nntrain) # save kl results for each nsim; same as kl.ave in previous versions
  kldiff.sum<-matrix(0, nrow=nnrepl, ncol=nntrain)
  klMean<-0

  eu.multi<-list()
  eudiff.multi<-list()
  eu.sum<-matrix(0, nrow=nnrepl, ncol=nntrain) # save eu results for each nsim; same as eu.ave in previous versions
  eudiff.sum<-matrix(0, nrow=nnrepl, ncol=nntrain)
  euMean<-0

  for (t in 1:nsim) {
    ############ Simulated data sets (N variables) ############
    # training data set
    sigma0<-diag(N)
    sigma1<-diag(N)
    sigma1[lower.tri(sigma1)]<-0.5
    sigma1[upper.tri(sigma1)]<-0.5
    train0<-mvrnorm(n0, rep(0,N), sigma0)
    train1<-mvrnorm(n1, rep(1,N), sigma1)
    train2<-mvrnorm(n2, rep(2,N), sigma0)
    # test data set
    test0<-mvrnorm(n0, rep(0,N), sigma0)
    test1<-mvrnorm(n1, rep(1,N), sigma1)
```

```
    test2<-mvrnorm(n2, rep(2,N), sigma0)
    # data sets summary
    train.X<-rbind(train0,train1,train2)
    test.X<-rbind(test0,test1,test2)
    train.y<-c(rep(0,n0),rep(1,n1),rep(2,n2))
    test.y<-c(rep(0,n0),rep(1,n1),rep(2,n2))
    factor.y<-as.factor(train.y)
    df.train<-as.data.frame(cbind(train.y,train.X))
    df.test<-as.data.frame(cbind(test.y,test.X))

    ##############################################
    ############# Original Assessmet #############
    ##############################################

    if (models=="knn") {
      ############ KKNN ############
      y.fit<-kknn(factor.y~., df.train, df.test, k=kValue)
      y.prob<-y.fit$"prob"
    } else if (models=="ld") {
      ############ Linear Discriminant ############
      y.fit<-lda(train.y~., data=df.train)
      y.prob<-predict(y.fit,df.test)$posterior
    } else if (models=="log") {
      ############ Logistic Regression ############
      y.fit<-multinom(factor.y~., data=df.train[,-1], trace=FALSE)
      y.prob<-predict(y.fit,df.test,type="probs")
    } else if (models=="svm") {
      ############ SVM ############
      y.fit<-svm(factor.y~., data=df.train[,-1], probability=TRUE)
      y.pred<-predict(y.fit,newdata=df.test,probability=TRUE)
      y.prob<-attr(y.pred,"probabilities")
    } else if (models=="dtree") {
      ############ Decision Tree ############
      y.fit<-tree(factor.y~., data=df.train[,-1])
      y.prob<-predict(y.fit, df.test, type="vector")
    } else if (models=="ptree") {
      ############ Prune Tree ############
      y.auto<-rpart(factor.y~., data=df.train[,-1])
      y.fit<-prune(y.auto, cp=0.1)
      y.prob<-predict(y.fit, df.test)
    } else if (models=="forest") {
      ############ Random Forest ############
      y.fit<-randomForest(factor.y~.,data=df.train[,-1],sampsize=train.size)
      y.prob<-predict(y.fit,newdata=df.test,type="prob")
    } else if (models=="neural") {
      ############ Neural Network ############
      y.fit<-nnet(factor.y~., data=df.train[,-1],size=2,decay = 5e-4, maxit = 200, trace=FALSE)
      y.prob<-predict(y.fit, df.test,type="raw")
    } else if (models=="nb") {
      ############ Naive Bayes ############
      y.fit<-naiveBayes(factor.y~.,data=df.train[,-1])
      y.prob=predict(y.fit,df.test,type = "raw")
```

```
      } else if (models=="C50") {
        ############ C50 ############
        y.fit<-C5.0(factor.y~., data=df.train[,-1],rules=FALSE)
        y.prob<-predict(y.fit, df.test, type = "prob")
      } else if (models=="fda") {
        ############ FDA ############
        y.fit<-fda(factor.y~., data=df.train[,-1])
        y.prob<-predict(y.fit, df.test,type = "posterior")
      } else if (models=="pls") {
        ############ PLS ############
        y.fit<-plsda(df.train[,-1],factor.y)
        y.prob<-predict(y.fit,df.test[,-1],type = "prob")[,1:(max(df.
    test[,1])+1),]
      } else if (models=="mda") {
        ############ MDA ############
        y.fit<-mda(factor.y~., data=df.train[,-1])
        y.prob<-predict(y.fit,df.test,type = "posterior")
      } else {
        stop("Wrong model type!")
        quit("no")  # not working?
      }

      # evaluation
      # roc
      roc0<-multiclass.roc(test.y, y.prob[,1])$auc
      # kl
      log.prob<-log(1/(y.prob+e))
      log.matrix<-cbind(df.test[,1],log.prob)
      kl.list<-list()
      # p - categories for df.test[,1]
      for (p in c(0:max(df.test[,1]))) {
        kl.list[[p+1]]<-sum(log.matrix[log.matrix[,1] == p,][,(p+2)])
      }
      kl0<-Reduce("+",kl.list)
      # eu
      eu0<-sum((test.y-y.prob)^2)

      rocMean<-rocMean+roc0
      klMean<-klMean+kl0
      euMean<-euMean+eu0
      # print (roc0)

      ###############################################
      ######## Vibration in Testing Data Set ########
      ###############################################

      # store the results of roc areas for each pair of training and
    validation data sets
      yprob.single<-matrix(0, nrow=nnrepl, ncol=nntrain)
      roc.table<-matrix(0, nrow=nnrepl, ncol=nntrain)
      roc.diff<-matrix(0, nrow=nnrepl, ncol=nntrain)

      kl.table<-matrix(0, nrow=nnrepl, ncol=nntrain)
      kl.diff<-matrix(0, nrow=nnrepl, ncol=nntrain)
```

```
    eu.table<-matrix(0, nrow=nnrepl, ncol=nntrain)
    eu.diff<-matrix(0, nrow=nnrepl, ncol=nntrain)
    ################################
    for (j in 1:nnrepl) {
      rare.size<-j*n1
      total.size<-rare.size+n0+n2 # size of the vibrated training data set: (n0+j*n1)
      train1.star<-train1[rep(seq_len(nrow(train1)), j), ] # duplicate the rare part
      varDiag<-diag(colVars(as.matrix(train1.star)))  # sample variance diagonal (sigma q)
      trainVib.y<-c(rep(0,n0),rep(1,rare.size),rep(2,n2))
      factoryVib.y<-as.factor(trainVib.y)
      yhat<-0
      for (i in 1:nntrain) {
        # add noise to each rare part in the training data set
        noise<-mvrnorm(rare.size, rep(0,N), k*diag(N), empirical = TRUE) # epsilson
        train1.vib<-train1.star+noise # vibrate the rare part
        train1.anti<-train1.star-noise # add anti-noise

        # generate the training data set with j rare parts (y=1)
        trainVib.X<-rbind(train0,train1.vib,train2) # training data set after vibrating the rare part
        trainVib<-as.data.frame(cbind(trainVib.y, trainVib.X))
        trainAnti.X<-rbind(train0,train1.anti,train2)
        trainAnti<-as.data.frame(cbind(trainVib.y, trainAnti.X))

        # models
        if (models=="knn") {
          ############ kknn ############
          y.pred_noise<-kknn(factoryVib.y~., trainVib, df.test, k=kValue)
          y.prob_noise<-y.pred_noise$"prob"

          y.pred_anti<-kknn(factoryVib.y~., trainAnti, df.test, k=kValue)
          y.prob_anti<-y.pred_anti$"prob"
        }  else if (models=="ld") {
          ############ lda ############
          lda.fit<-lda(trainVib.y~., data=trainVib[,-1])
          y.prob_noise<-predict(lda.fit,df.test)$posterior

          lda.fit<-lda(trainVib.y~., data=trainAnti[,-1])
          y.prob_anti<-predict(lda.fit,df.test)$posterior
        } else if (models=="log") {
          ############ Logistic Regression ############
          y.fit<-multinom(factoryVib.y~., data=trainVib[,-1], trace=FALSE)
          y.prob_noise<-predict(y.fit,df.test,type="probs")

          y.fit<-multinom(factoryVib.y~., data=trainAnti[,-1], trace=FALSE)
```

```
            y.prob_anti<-predict(y.fit,df.test,type="probs")
        } else if (models=="svm") {
            ############ SVM ############
            y.fit<-svm(factoryVib.y~., data=trainVib[,-1], probability=
    TRUE)
            y.pred<-predict(y.fit,newdata=df.test,probability=TRUE)
            y.prob_noise<-attr(y.pred,"probabilities")

            y.fit<-svm(factoryVib.y~., data=trainAnti[,-1], probability=
    TRUE)
            y.pred<-predict(y.fit,newdata=df.test,probability=TRUE)
            y.prob_anti<-attr(y.pred,"probabilities")
        } else if (models=="dtree") {
            ############ Decision Tree ############
            y.fit<-tree(factoryVib.y~., data=trainVib[,-1])
            y.prob_noise<-predict(y.fit, df.test, type="vector")

            y.fit<-tree(factoryVib.y~., data=trainAnti[,-1])
            y.prob_anti<-predict(y.fit, df.test, type="vector")
        } else if (models=="ptree") {
            ############ Prune Tree ############
            y.auto<-rpart(factoryVib.y~., data=trainVib[,-1])
            y.fit<-prune(y.auto, cp=0.1)
            y.prob_noise<-predict(y.fit, df.test)

            y.auto<-rpart(factoryVib.y~., data=trainAnti[,-1])
            y.fit<-prune(y.auto, cp=0.1)
            y.prob_anti<-predict(y.fit, df.test)
        } else if (models=="forest") {
            ############ Random Forest ############
            y.fit<-randomForest(factoryVib.y~., data=trainVib[,-1],
    sampsize=train.size)
            y.prob_noise<-predict(y.fit,newdata=df.test,type="prob")

            y.fit<-randomForest(factoryVib.y~., data=trainAnti[,-1],
    sampsize=train.size)
            y.prob_anti<-predict(y.fit,newdata=df.test,type="prob")
        } else if (models=="neural") {
            ############ Neural Network ############
            y.fit<-nnet(factoryVib.y~., data=trainVib[,-1],size=2,decay
    = 5e-4, maxit = 200, trace=FALSE)
            y.prob_noise<-predict(y.fit, df.test,type="raw")

            y.fit<-nnet(factoryVib.y~., data=trainAnti[,-1],size=2,decay
     = 5e-4, maxit = 200, trace=FALSE)
            y.prob_anti<-predict(y.fit, df.test,type="raw")
        } else if (models=="nb") {
            ############ Naive Bayes ############
            y.fit<-naiveBayes(factoryVib.y~.,data=trainVib[,-1])
            y.prob_noise<-predict(y.fit,df.test,type = "raw")

            y.fit<-naiveBayes(factoryVib.y~.,data=trainAnti[,-1])
            y.prob_anti<-predict(y.fit,df.test,type = "raw")
        } else if (models=="C50") {
```

```
      ############ C50 ############
      y.fit<-C5.0(factoryVib.y~., data=trainVib[,-1],rules=TRUE)
      y.prob_noise<-predict(y.fit, df.test, type = "prob")

      y.fit<-C5.0(factoryVib.y~., data=trainAnti[,-1],rules=TRUE)
      y.prob_anti<-predict(y.fit, df.test, type = "prob")
    } else if (models=="fda") {
      ############ FDA ############
      y.fit<-fda(factoryVib.y~., data=trainVib[,-1])
      y.prob_noise<-predict(y.fit, df.test, type = "posterior")

      y.fit<-fda(factoryVib.y~., data=trainAnti[,-1])
      y.prob_anti<-predict(y.fit, df.test, type = "posterior")
    } else if (models=="pls") {
      ############ PLS ############
      y.fit<-plsda(trainVib[,-1],factoryVib.y)
      y.prob_noise<-predict(y.fit, df.test[,-1], type = "prob")
[,1:(max(df.test[,1])+1),]

      y.fit<-plsda(trainAnti[,-1],factoryVib.y)
      y.prob_anti<-predict(y.fit, df.test[,-1], type = "prob")
[,1:(max(df.test[,1])+1),]
    } else if (models=="mda") {
      ############ MDA ############
      y.fit<-mda(factoryVib.y~., data=trainVib[,-1])
      y.prob_noise<-predict(y.fit, df.test, type = "posterior")

      y.fit<-mda(factoryVib.y~., data=trainAnti[,-1])
      y.prob_anti<-predict(y.fit, df.test, type = "posterior")
    } else {
      stop("Wrong model type! Please use lower cases")
    }

    # prediction probabilities after two-size vibration
    yhat<-yhat+((y.prob_noise+y.prob_anti)/2)   # accumulative
yhat

    # assessment
    y.prob<-yhat/i
    roc.table[j,i]<-multiclass.roc(test.y, y.prob[,1])$auc        #
 final roc table, same size as roc.summary
    roc.diff[j,i]<-roc.table[j,i]-roc0

    #kl
    log.prob<-log(1/(y.prob+e))
    log.matrix<-cbind(df.test[,1],log.prob)
    kl.list<-list()
    # p - categories for df.test[,1]
    for (p in c(0:max(df.test[,1]))) {
      kl.list[[p+1]]<-sum(log.matrix[log.matrix[,1] == p,][,(p+2)
])
    }
    kl.table[j,i]<-Reduce("+",kl.list)
    kl.diff[j,i]<-kl.table[j,i]-kl0
```

```
        #eu
        eu.table[j,i]<-sum((test.y-y.prob)^2)       # final eu table, same size as eu.summary
        eu.diff[j,i]<-eu.table[j,i]-eu0
      }
      plot(roc.table[j,])
      abline(h = roc0)

      plot(kl.table[j,])
      abline(h = kl0)

      plot(eu.table[j,])
      abline(h = eu0)
    }
    roc.multi[[t]]<-roc.table
    roc.sum<-roc.sum+roc.table
    rocdiff.multi[[t]]<-roc.diff
    rocdiff.sum<-rocdiff.sum+roc.diff

    kl.multi[[t]]<-kl.table
    kl.sum<-kl.sum+kl.table
    kldiff.multi[[t]]<-kl.diff
    kldiff.sum<-kldiff.sum+kl.diff

    eu.multi[[t]]<-eu.table
    eu.sum<-eu.sum+eu.table
    eudiff.multi[[t]]<-eu.diff
    eudiff.sum<-eudiff.sum+eu.diff
  }

  cat("\nOriginal ROC Mean =", rocMean/nsim, "\n")

  # roc final result
  roc.final<-(roc.sum/nsim)[noisy.repl,noisy.train]
  print ("roc results: ")
  print (roc.final)

  # roc difference
  rocdiff.final<-(rocdiff.sum/nsim)[noisy.repl,noisy.train]
  print ("roc difference: ")
  print (rocdiff.final)

  # plot roc difference (noisy.repl=2, nosiy.train=10)
  roc.ci.table<-c()
  for (t in 1:nsim) {
    roc.ci.table<-append(roc.ci.table,rocdiff.multi[[t]][nnrepl,nntrain])
  }
  roc.diff.ci<-append(roc.diff.ci,(qnorm(0.975)*sd(roc.ci.table)/sqrt(nsim)))    # roc difference ci
  roc.diff.mean<-append(roc.diff.mean, (rocdiff.sum/nsim)[nnrepl,nntrain])     # roc difference mean
```

```
    ##################################
    cat("\nOriginal KL Mean =", klMean/nsim, "\n")

    # kl final result
    kl.final<-(kl.sum/nsim)[noisy.repl,noisy.train]
    print ("kl results: ")
    print (kl.final)

    # kl difference
    kldiff.final<-(kldiff.sum/nsim)[noisy.repl,noisy.train]
    print ("kl difference: ")
    print (kldiff.final)

    # plot kl difference (noisy.repl=2, nosiy.train=10)
    kl.ci.table<-c()
    for (t in 1:nsim) {
      kl.ci.table<-append(kl.ci.table,kldiff.multi[[t]][nnrepl,nntrain])
    }
    kl.diff.ci<-append(kl.diff.ci,(qnorm(0.975)*sd(kl.ci.table)/sqrt(
   nsim)))    # kl difference ci
    kl.diff.mean<-append(kl.diff.mean, (kldiff.sum/nsim)[nnrepl,nntrain
   ])     # kl difference mean

    ##################################
    cat("\nOriginal EU Mean =", euMean/nsim, "\n")

    # eu final result
    eu.final<-(eu.sum/nsim)[noisy.repl,noisy.train]
    print ("eu results: ")
    print (eu.final)

    # eu difference
    eudiff.final<-(eudiff.sum/nsim)[noisy.repl,noisy.train]
    print ("eu difference: ")
    print (eudiff.final)

    # plot eu difference (noisy.repl=2, nosiy.train=10)
    eu.ci.table<-c()
    for (t in 1:nsim) {
      eu.ci.table<-append(eu.ci.table,eudiff.multi[[t]][nnrepl,nntrain])
    }
    eu.diff.ci<-append(eu.diff.ci,(qnorm(0.975)*sd(eu.ci.table)/sqrt(
   nsim)))    # eu difference ci
    eu.diff.mean<-append(eu.diff.mean, (eudiff.sum/nsim)[nnrepl,nntrain
   ])     # eu difference mean
  }

 # plot roc difference among sigma.noise=(0.1,0.5,1.0)
 print("**********************")
 print("ROC diff mean:")
 print(roc.diff.mean)
 print("ROC diff CI")
 print(roc.diff.ci)
```

```
roc.plot<-matrix(0, nrow = 3, ncol = 3)
colnames(roc.plot)<-c("noise","mean","sd")
roc.plot[,1]<-c(0.1,0.5,1.0)
roc.plot[1,2:3]<-c(roc.diff.mean[1],roc.diff.ci[1])
roc.plot[2,2:3]<-c(roc.diff.mean[2],roc.diff.ci[2])
roc.plot[3,2:3]<-c(roc.diff.mean[3],roc.diff.ci[3])
roc.plot<-data.frame(noise=c(0.1,0.5,1.0),
                     mean=roc.plot[,2],
                     sd=roc.plot[,3])
p<-ggplot(roc.plot, aes(x=noise, y=mean), colour=mean) +
  geom_errorbar(aes(ymin=mean-sd, ymax=mean+sd), width=.1) +
  geom_line() +
  geom_point() +
  xlab("noise") +
  ylab("roc.diff") +
  geom_hline(yintercept = 0)

if (models=="knn") {
  roc.p1<-p+ggtitle("KNN ROC diff")
} else if (models=="ld") {
  roc.p2<-p+ggtitle("LD ROC diff")
} else if (models=="log") {
  roc.p3<-p+ggtitle("LOG ROC diff")
} else if (models=="svm") {
  roc.p4<-p+ggtitle("SVM ROC diff")
} else if (models=="dtree") {
  roc.p5<-p+ggtitle("Decision Tree ROC diff")
} else if (models=="ptree") {
  roc.p6<-p+ggtitle("Prune Tree ROC diff")
} else if (models=="forest") {
  roc.p7<-p+ggtitle("Random Forest ROC diff")
} else if (models=="neural") {
  roc.p8<-p+ggtitle("Neural Network ROC diff")
} else if (models=="nb") {
  roc.p9<-p+ggtitle("Naive Bayes ROC diff")
} else if (models=="C50") {
  roc.p10<-p+ggtitle("C5.0 ROC diff")
} else if (models=="fda") {
  roc.p11<-p+ggtitle("FDA ROC diff")
} else if (models=="pls") {
  roc.p12<-p+ggtitle("PLSDA ROC diff")
} else if (models=="mda") {
  roc.p13<-p+ggtitle("MDA ROC diff")
} else {
  stop("Wrong model type!!!")
}


##################################################
# plot kl difference among sigma.noise=(0.1,0.5,1.0)
print("**********************")
print("KL diff mean:")
print(kl.diff.mean)
```

```
print("KL diff CI")
print(kl.diff.ci)

kl.plot<-matrix(0, nrow = 3, ncol = 3)
colnames(kl.plot)<-c("noise","mean","sd")
kl.plot[,1]<-c(0.1,0.5,1.0)
kl.plot[1,2:3]<-c(kl.diff.mean[1],kl.diff.ci[1])
kl.plot[2,2:3]<-c(kl.diff.mean[2],kl.diff.ci[2])
kl.plot[3,2:3]<-c(kl.diff.mean[3],kl.diff.ci[3])
kl.plot<-data.frame(noise=c(0.1,0.5,1.0),
                    mean=kl.plot[,2],
                    sd=kl.plot[,3])
p<-ggplot(kl.plot, aes(x=noise, y=mean), colour=mean) +
  geom_errorbar(aes(ymin=mean-sd, ymax=mean+sd), width=.1) +
  geom_line() +
  geom_point() +
  xlab("noise") +
  ylab("kl.diff") +
  geom_hline(yintercept = 0)

if (models=="knn") {
  kl.p1<-p+ggtitle("KNN kl diff")
} else if (models=="ld") {
  kl.p2<-p+ggtitle("LD kl diff")
} else if (models=="log") {
  kl.p3<-p+ggtitle("LOG kl diff")
} else if (models=="svm") {
  kl.p4<-p+ggtitle("SVM kl diff")
} else if (models=="dtree") {
  kl.p5<-p+ggtitle("Decision Tree kl diff")
} else if (models=="ptree") {
  kl.p6<-p+ggtitle("Prune Tree kl diff")
} else if (models=="forest") {
  kl.p7<-p+ggtitle("Random Forest kl diff")
} else if (models=="neural") {
  kl.p8<-p+ggtitle("Neural Network kl diff")
} else if (models=="nb") {
  kl.p9<-p+ggtitle("Naive Bayes kl diff")
} else if (models=="C50") {
  kl.p10<-p+ggtitle("C5.0 kl diff")
} else if (models=="fda") {
  kl.p11<-p+ggtitle("FDA kl diff")
} else if (models=="pls") {
  kl.p12<-p+ggtitle("PLSDA kl diff")
} else if (models=="mda") {
  kl.p13<-p+ggtitle("MDA kl diff")
} else {
  stop("Wrong model type!!!")
}


##################################################
# plot eu difference among sigma.noise=(0.1,0.5,1.0)
print("**********************")
```

```
  print("EU diff mean:")
  print(eu.diff.mean)
  print("EU diff CI")
  print(eu.diff.ci)

  eu.plot<-matrix(0, nrow = 3, ncol = 3)
  colnames(eu.plot)<-c("noise","mean","sd")
  eu.plot[,1]<-c(0.1,0.5,1.0)
  eu.plot[1,2:3]<-c(eu.diff.mean[1],eu.diff.ci[1])
  eu.plot[2,2:3]<-c(eu.diff.mean[2],eu.diff.ci[2])
  eu.plot[3,2:3]<-c(eu.diff.mean[3],eu.diff.ci[3])
  eu.plot<-data.frame(noise=c(0.1,0.5,1.0),
                      mean=eu.plot[,2],
                      sd=eu.plot[,3])
  p<-ggplot(eu.plot, aes(x=noise, y=mean), colour=mean) +
    geom_errorbar(aes(ymin=mean-sd, ymax=mean+sd), width=.1) +
    geom_line() +
    geom_point() +
    xlab("noise") +
    ylab("eu.diff") +
    geom_hline(yintercept = 0)

  if (models=="knn") {
    eu.p1<-p+ggtitle("KNN eu diff")
  } else if (models=="ld") {
    eu.p2<-p+ggtitle("LD eu diff")
  } else if (models=="log") {
    eu.p3<-p+ggtitle("LOG eu diff")
  } else if (models=="svm") {
    eu.p4<-p+ggtitle("SVM eu diff")
  } else if (models=="dtree") {
    eu.p5<-p+ggtitle("Decision Tree eu diff")
  } else if (models=="ptree") {
    eu.p6<-p+ggtitle("Prune Tree eu diff")
  } else if (models=="forest") {
    eu.p7<-p+ggtitle("Random Forest eu diff")
  } else if (models=="neural") {
    eu.p8<-p+ggtitle("Neural Network eu diff")
  } else if (models=="nb") {
    eu.p9<-p+ggtitle("Naive Bayes eu diff")
  } else if (models=="C50") {
    eu.p10<-p+ggtitle("C5.0 eu diff")
  } else if (models=="fda") {
    eu.p11<-p+ggtitle("FDA eu diff")
  } else if (models=="pls") {
    eu.p12<-p+ggtitle("PLSDA eu diff")
  } else if (models=="mda") {
    eu.p13<-p+ggtitle("MDA eu diff")
  } else {
    stop("Wrong model type!!!")
  }
}
```

```
multiplot(roc.p1, roc.p2, roc.p3, roc.p4, roc.p5, roc.p6, roc.p8, roc.p9
    , roc.p10, roc.p11, roc.p12, roc.p13, cols=6)
multiplot(kl.p1, kl.p2, kl.p3, kl.p4, kl.p5, kl.p6, kl.p8, kl.p9, kl.p10
    , kl.p11, kl.p12, kl.p13, cols=6)
multiplot(eu.p1, eu.p2, eu.p3, eu.p4, eu.p5, eu.p6, eu.p8, eu.p9, eu.p10
    , eu.p11, eu.p12, eu.p13, cols=6)

# running time
proc.time() - ptm
```

Listing 1: R code example

B Outcomes for Binary Data Sets

This section contains outcomes for selected binary data sets in testing the noisy replication method. They are ordered by the imbalance ratio (IR). The first subgraph in each figure is the 95% confident intervals of $\triangle$ROC, and the second is the 95% confident intervals of $\triangle$KL distance after applying the noisy replication method.

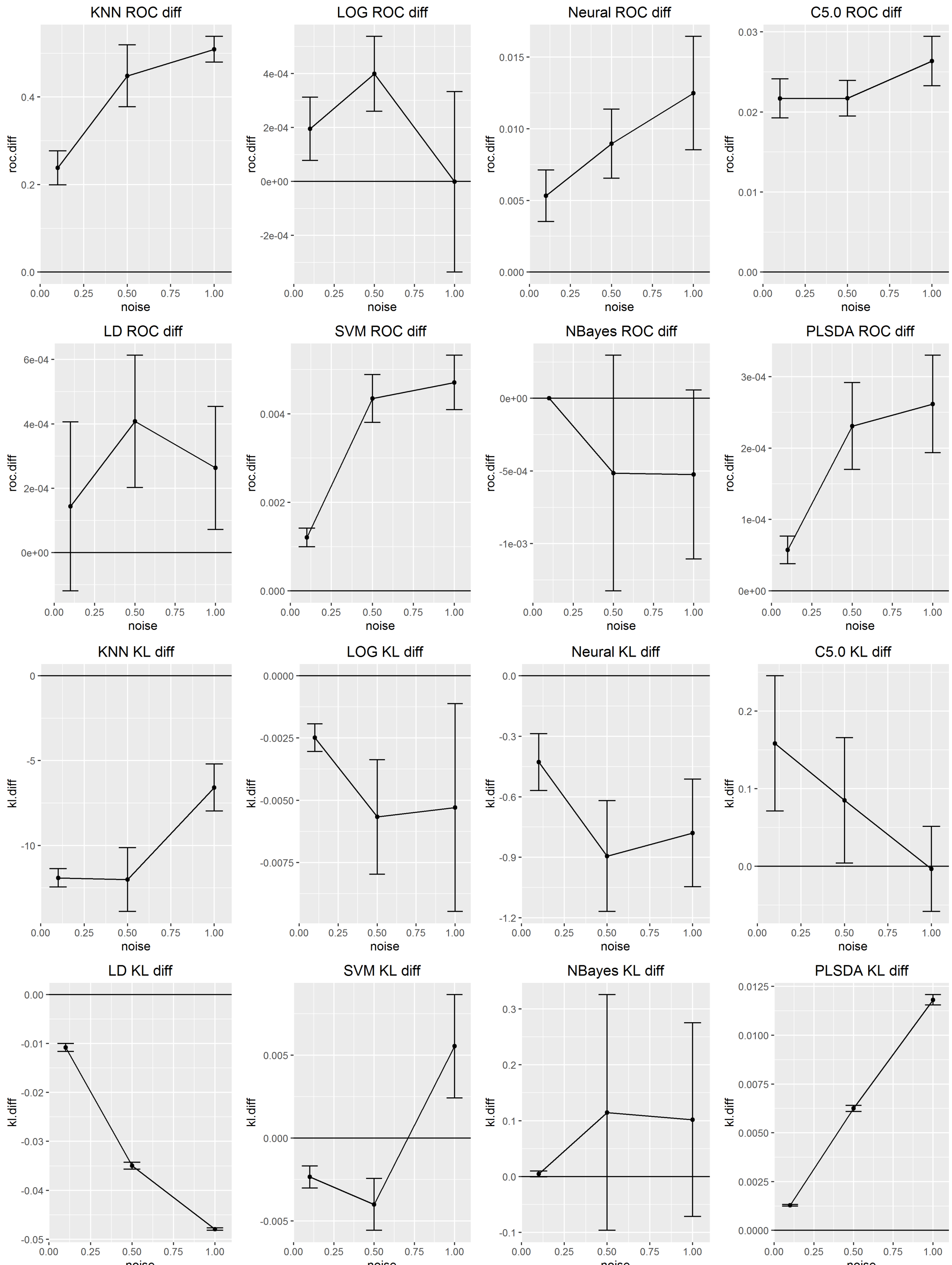


Figure 1: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 1.86

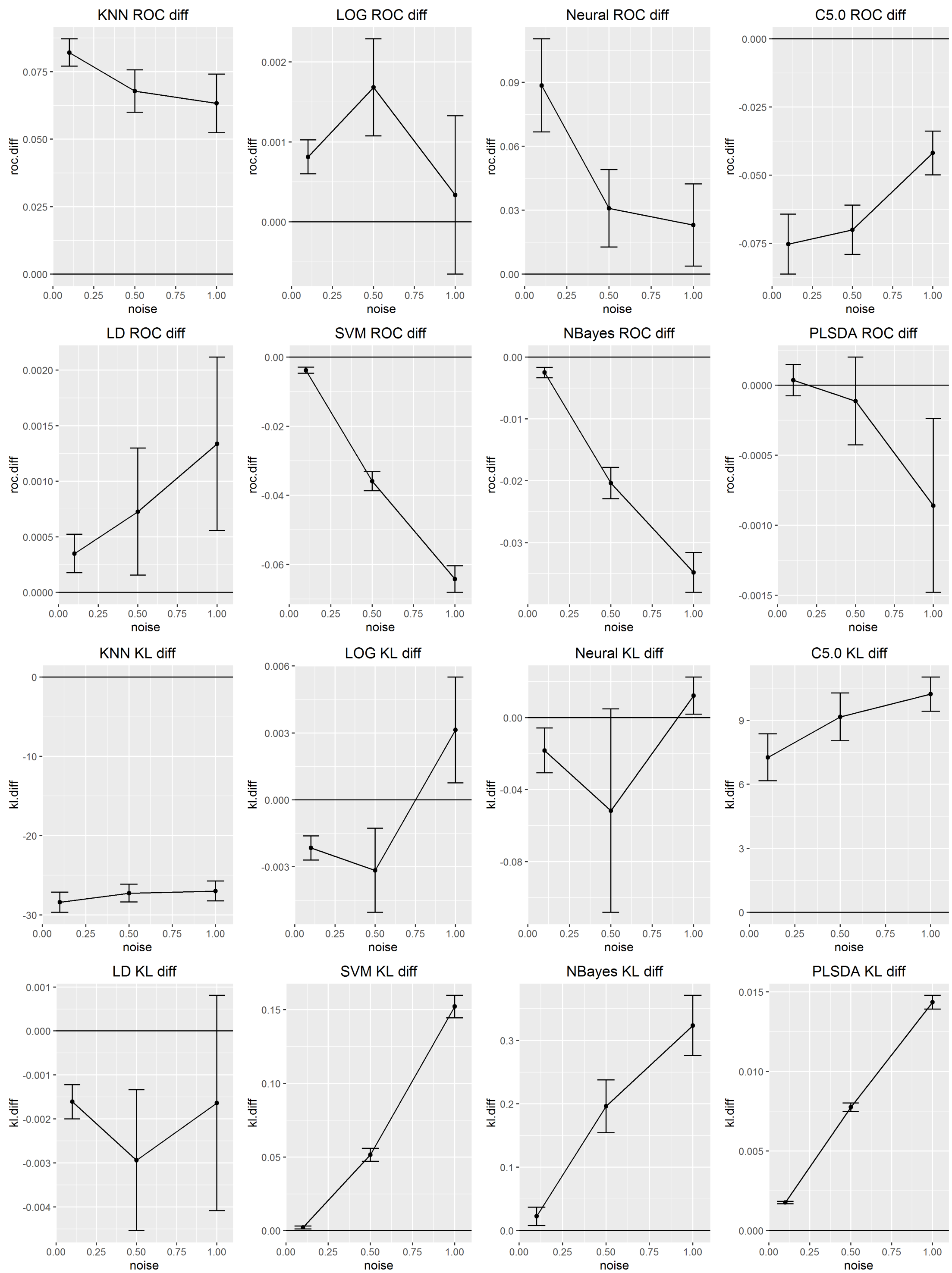


Figure 2: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 1.87

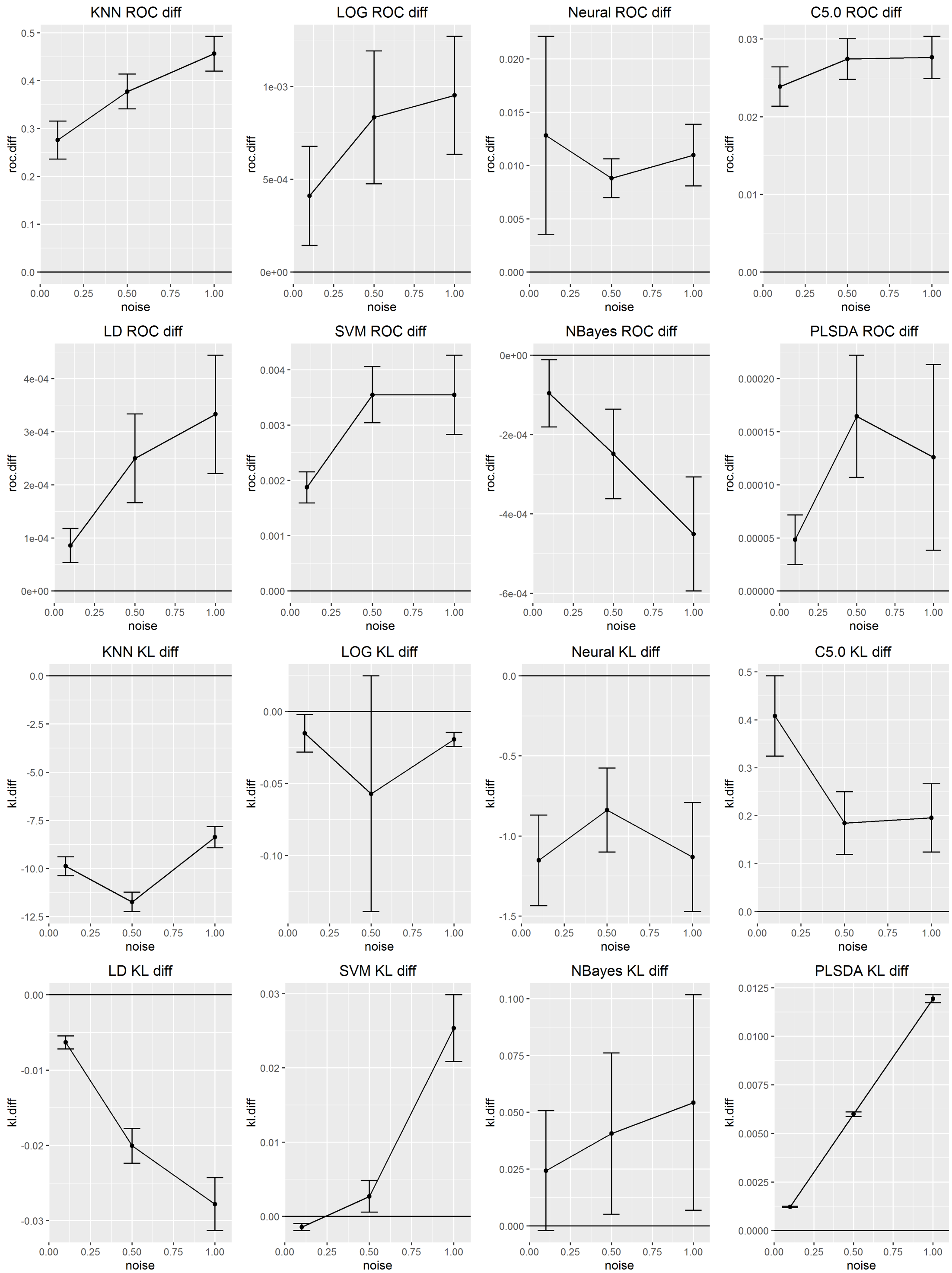


Figure 3: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 1.90

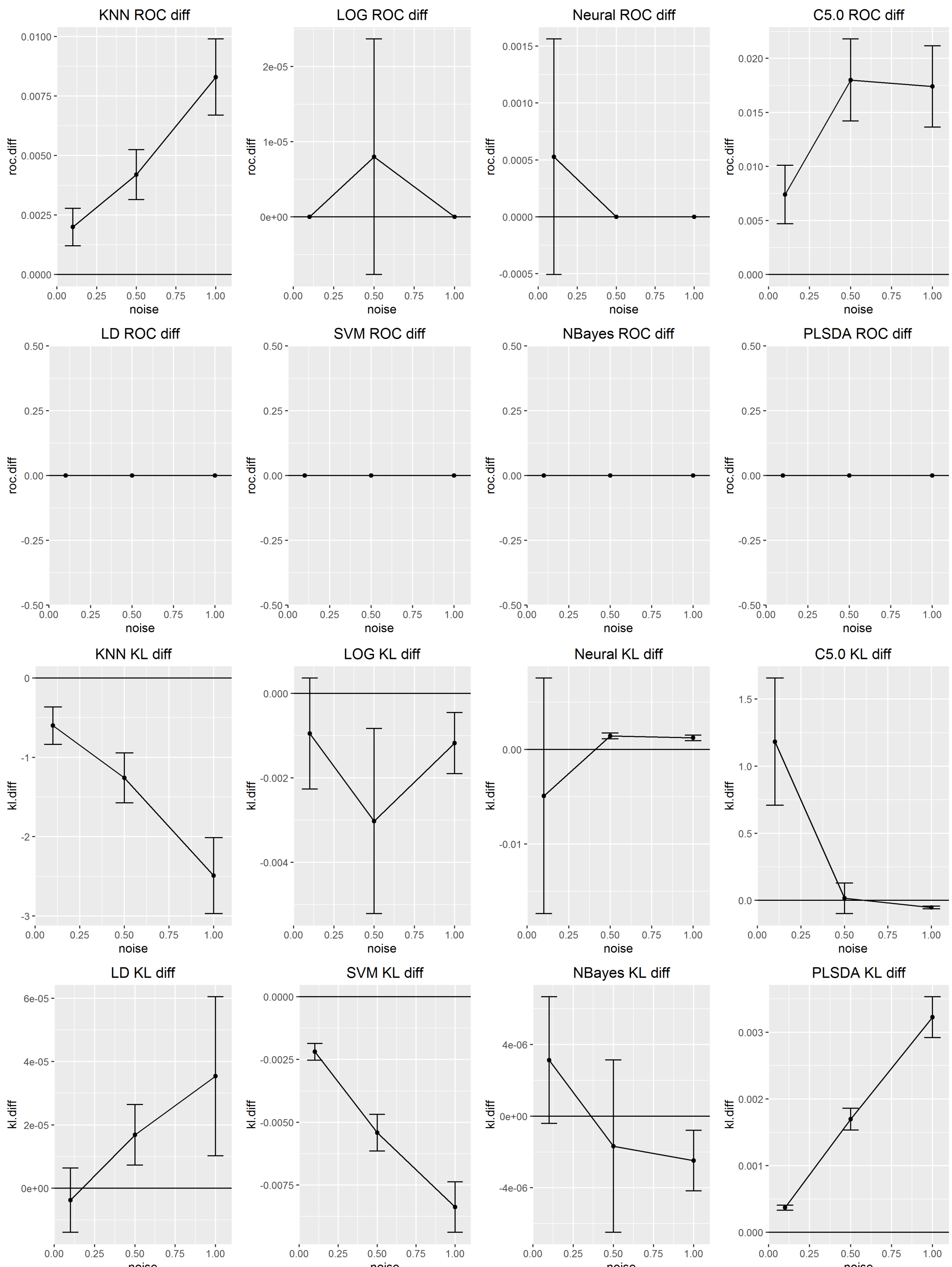


Figure 4: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 2.00

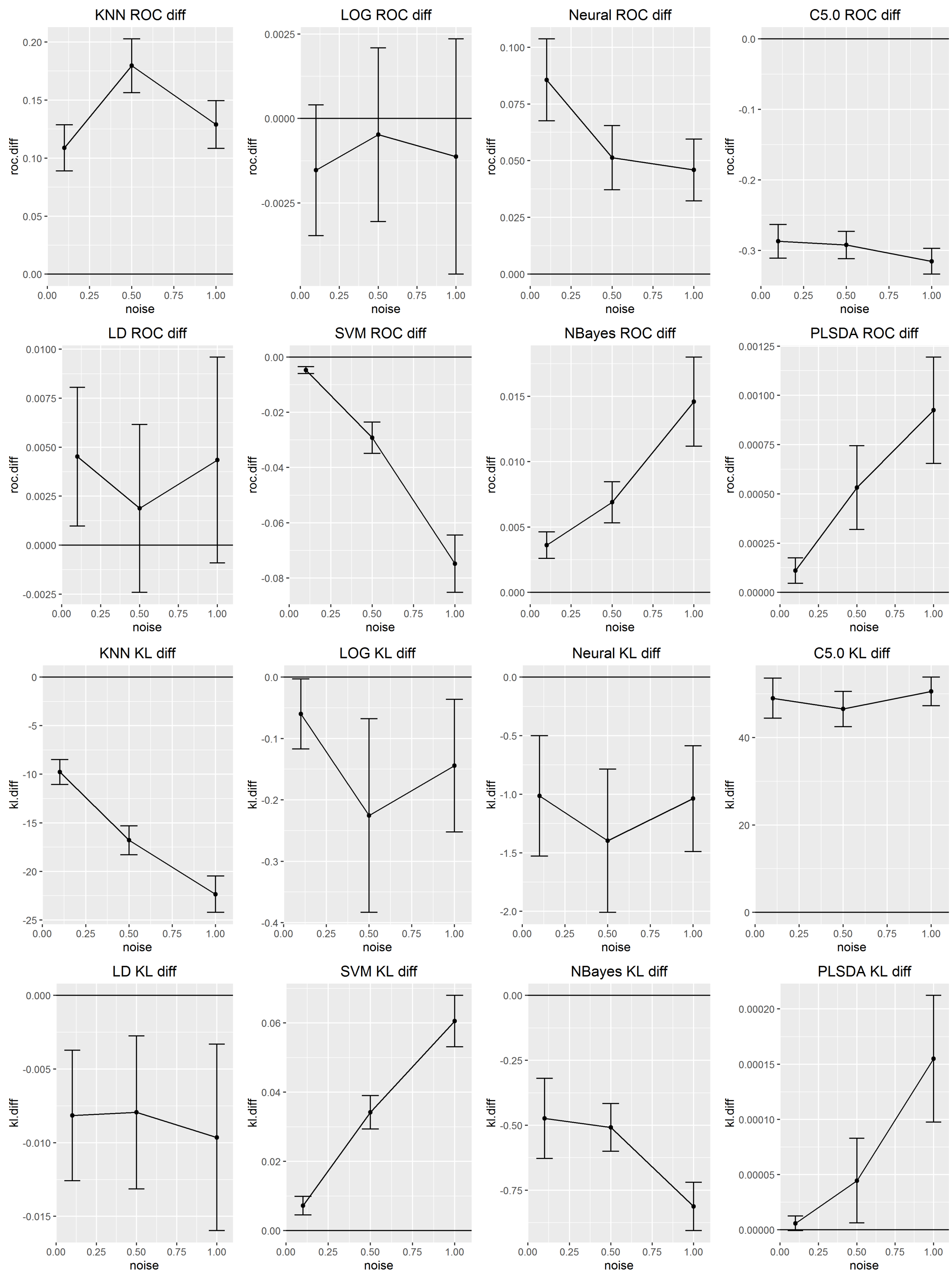


Figure 5: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 2.06

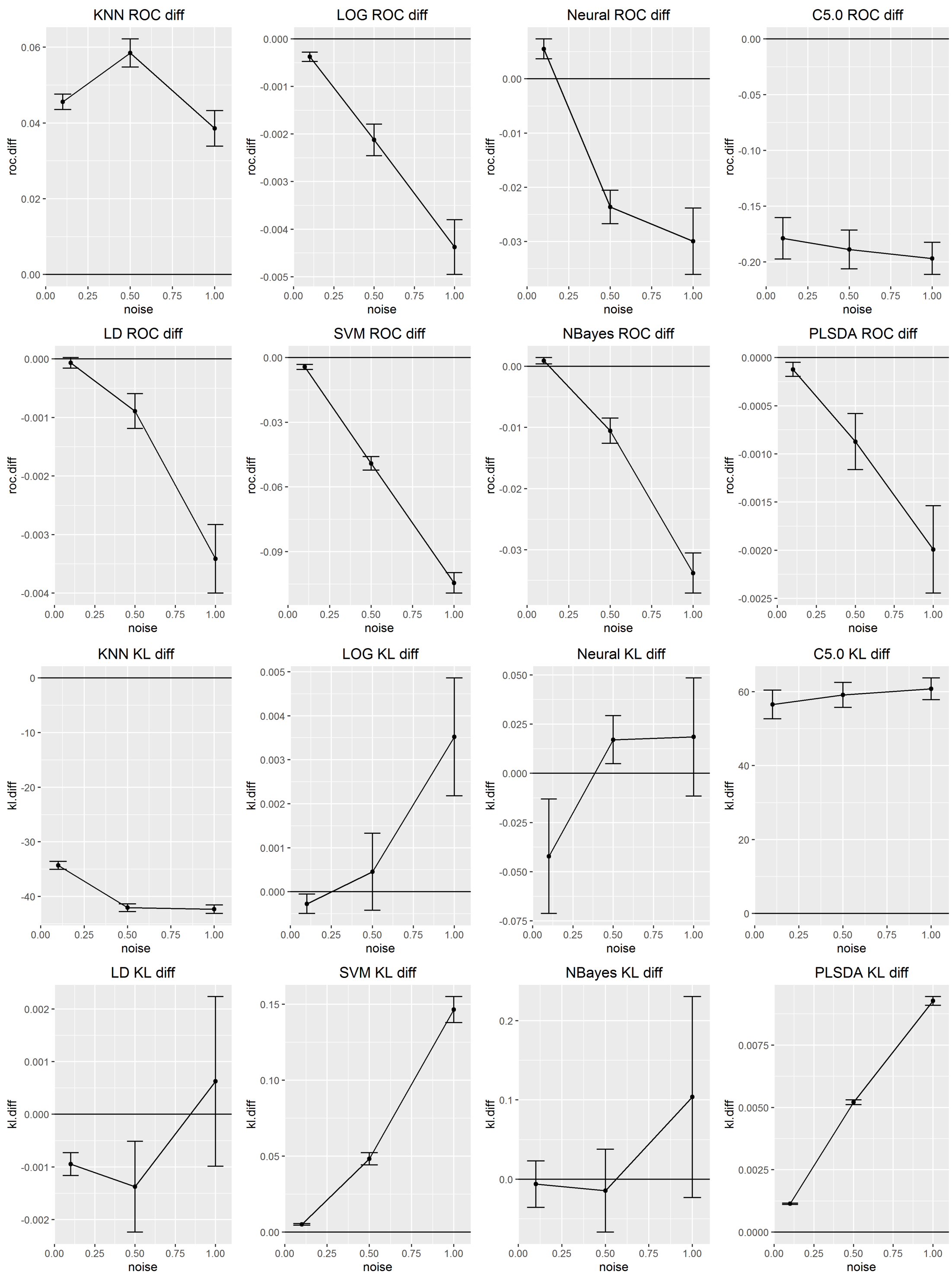


Figure 6: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 2.46

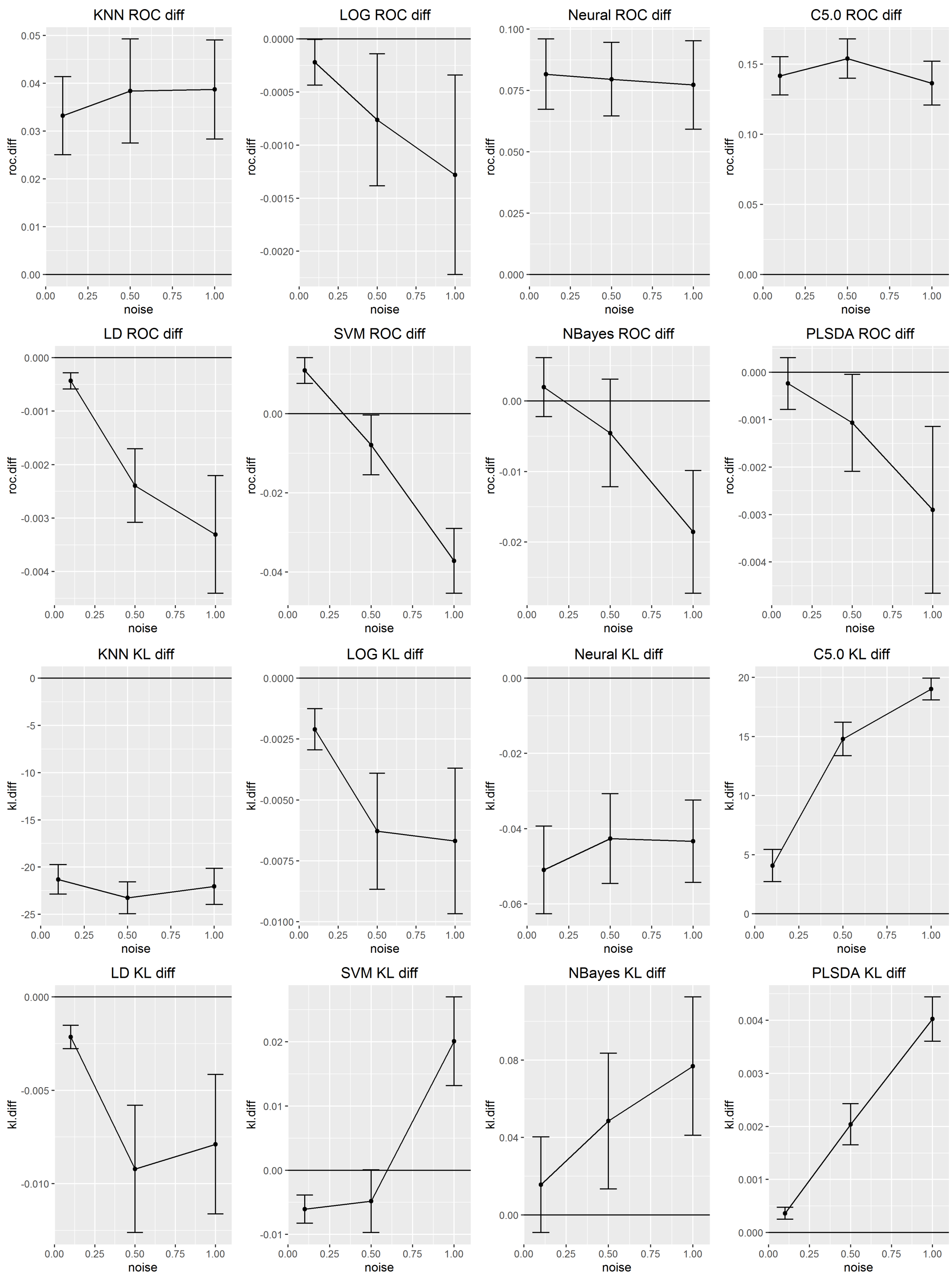


Figure 7: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 2.78

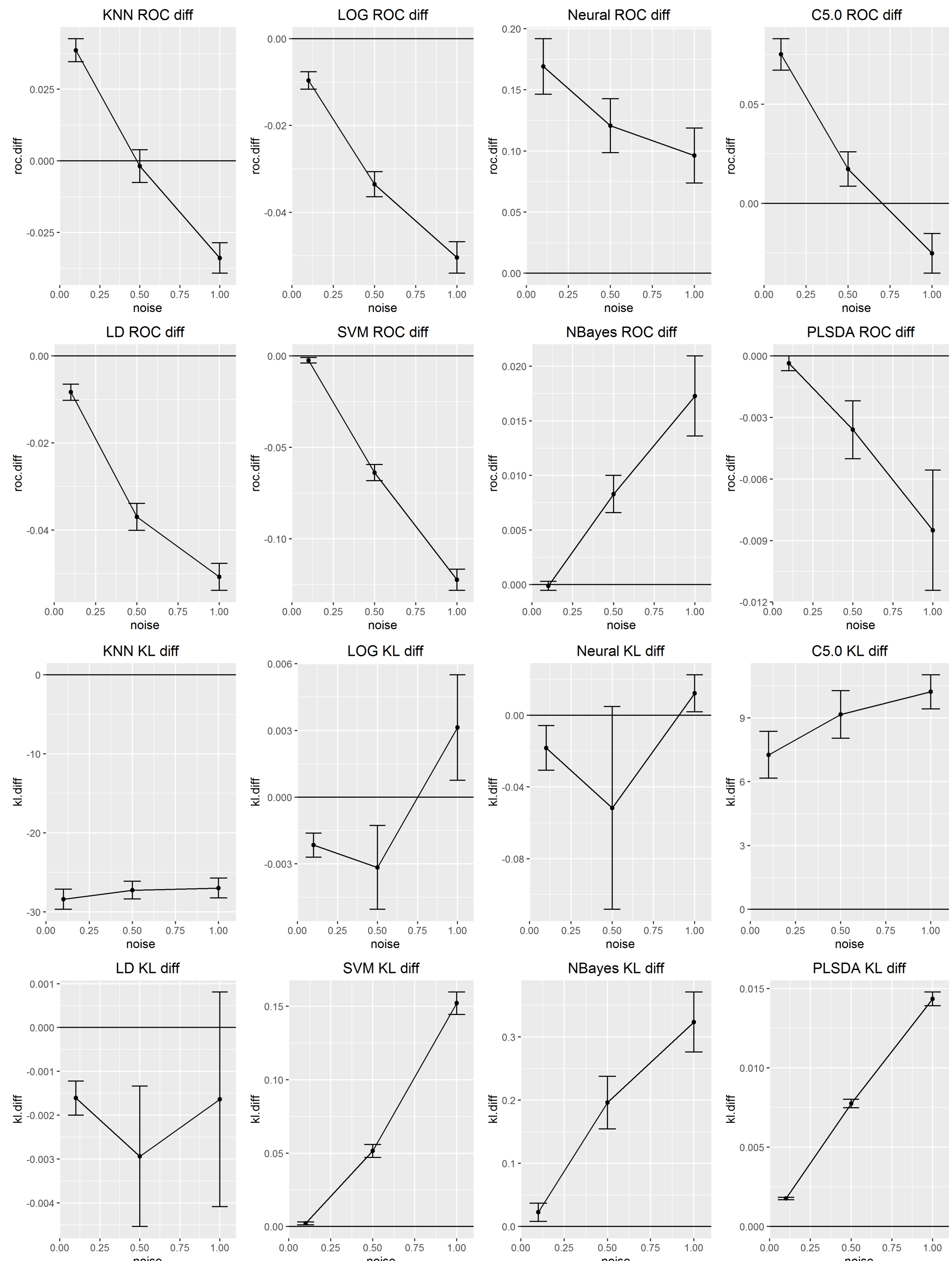


Figure 8: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 2.99

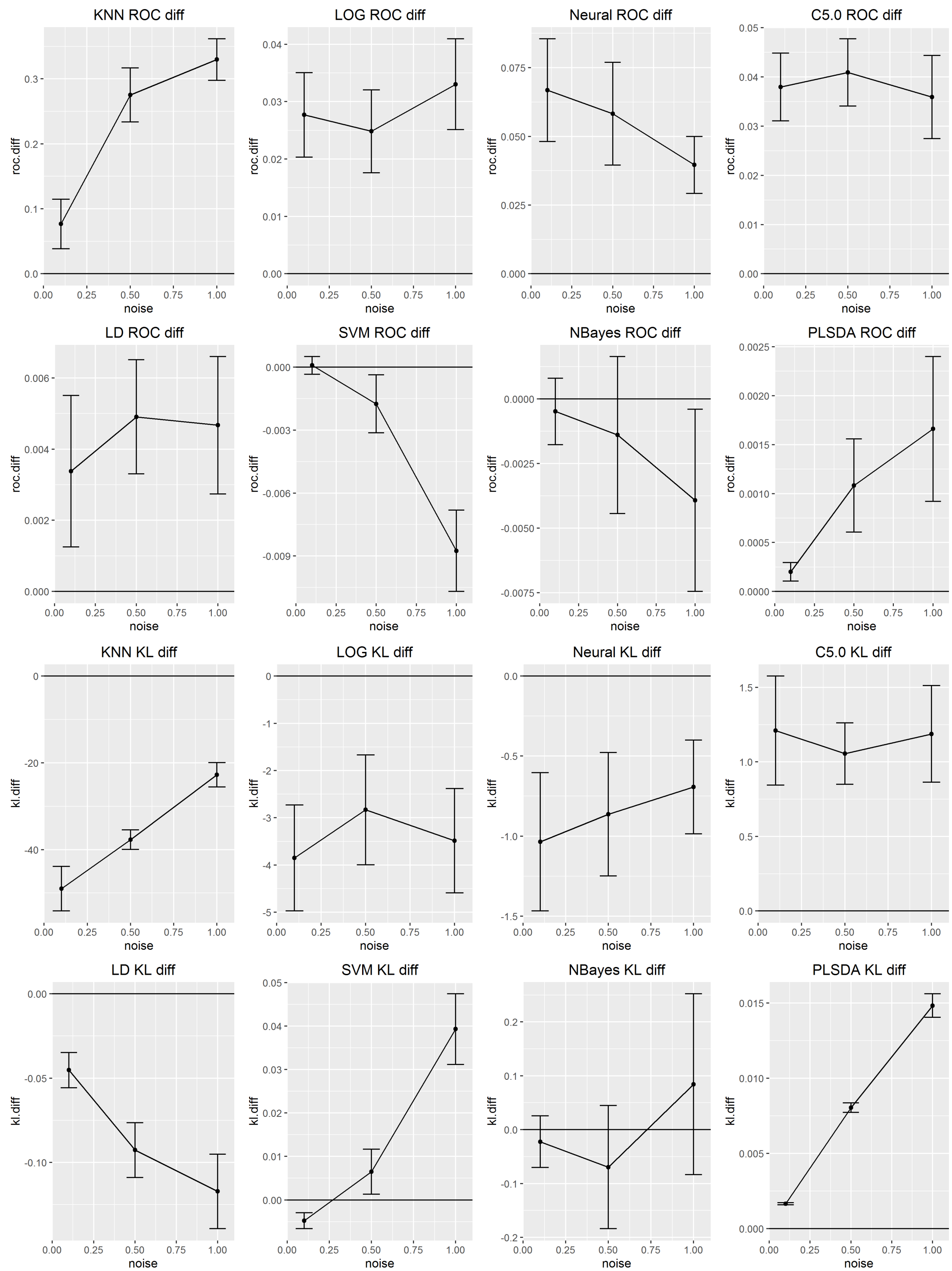


Figure 9: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 3.20

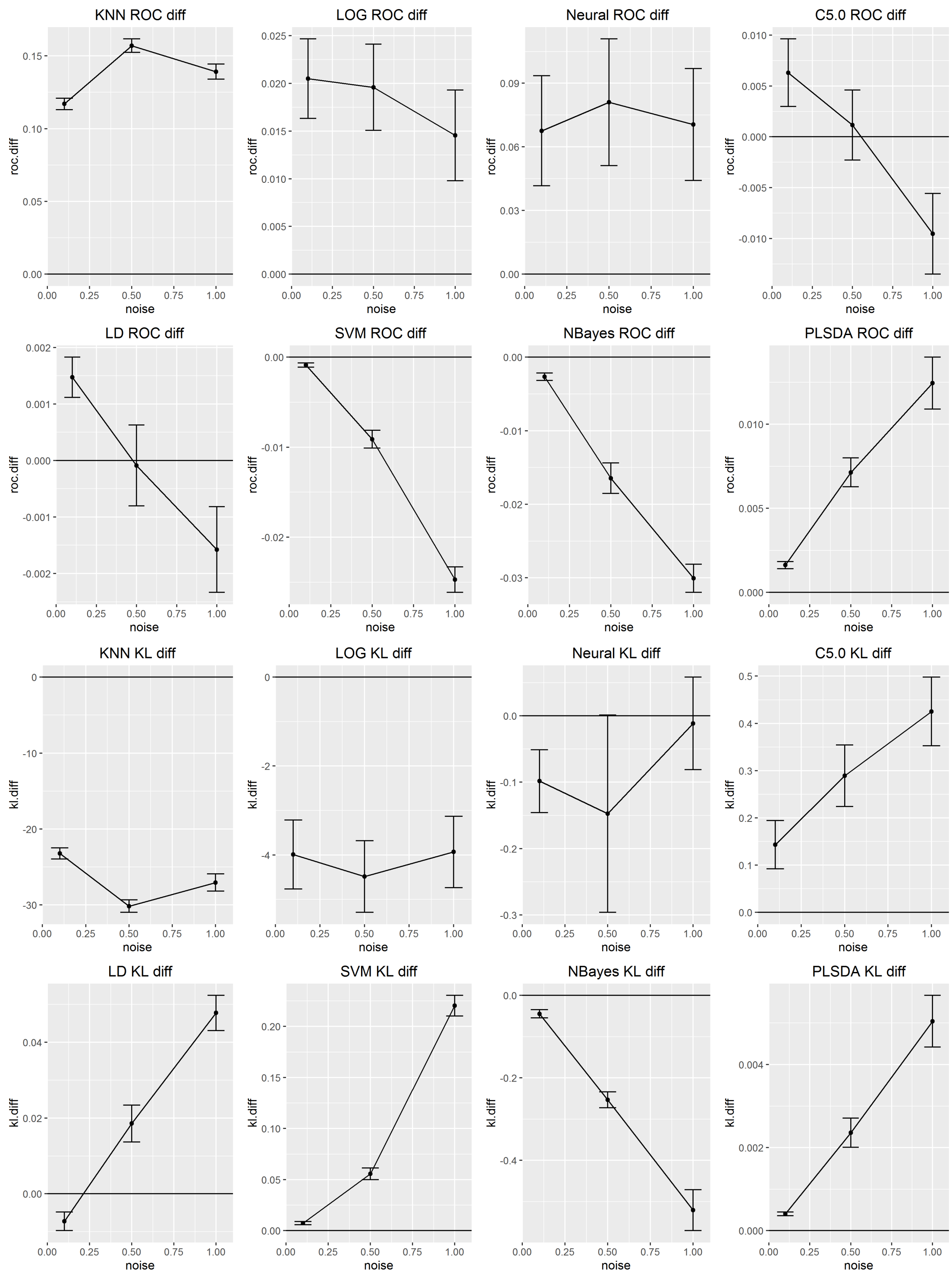


Figure 10: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 3.25

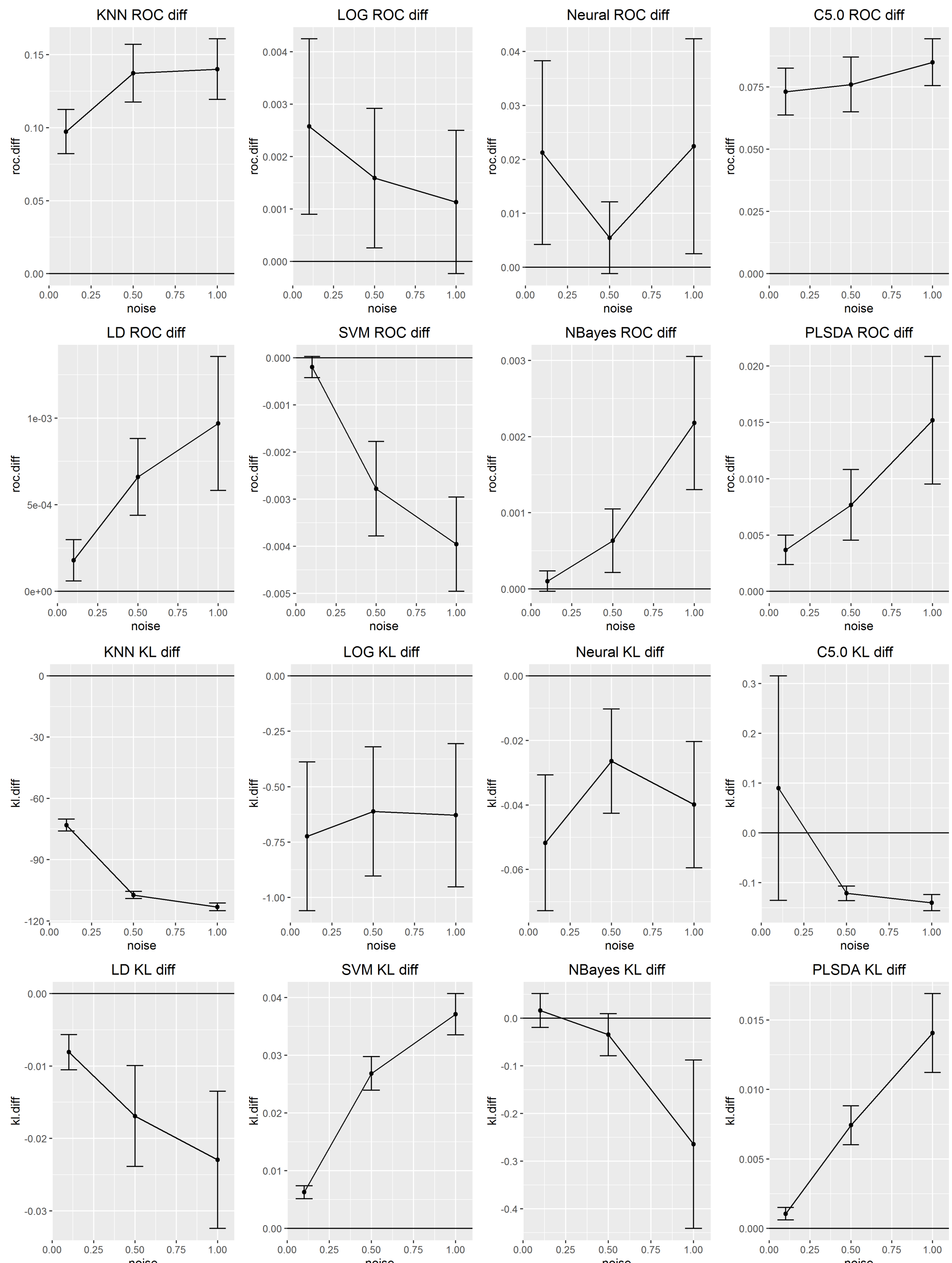


Figure 11: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 5.14

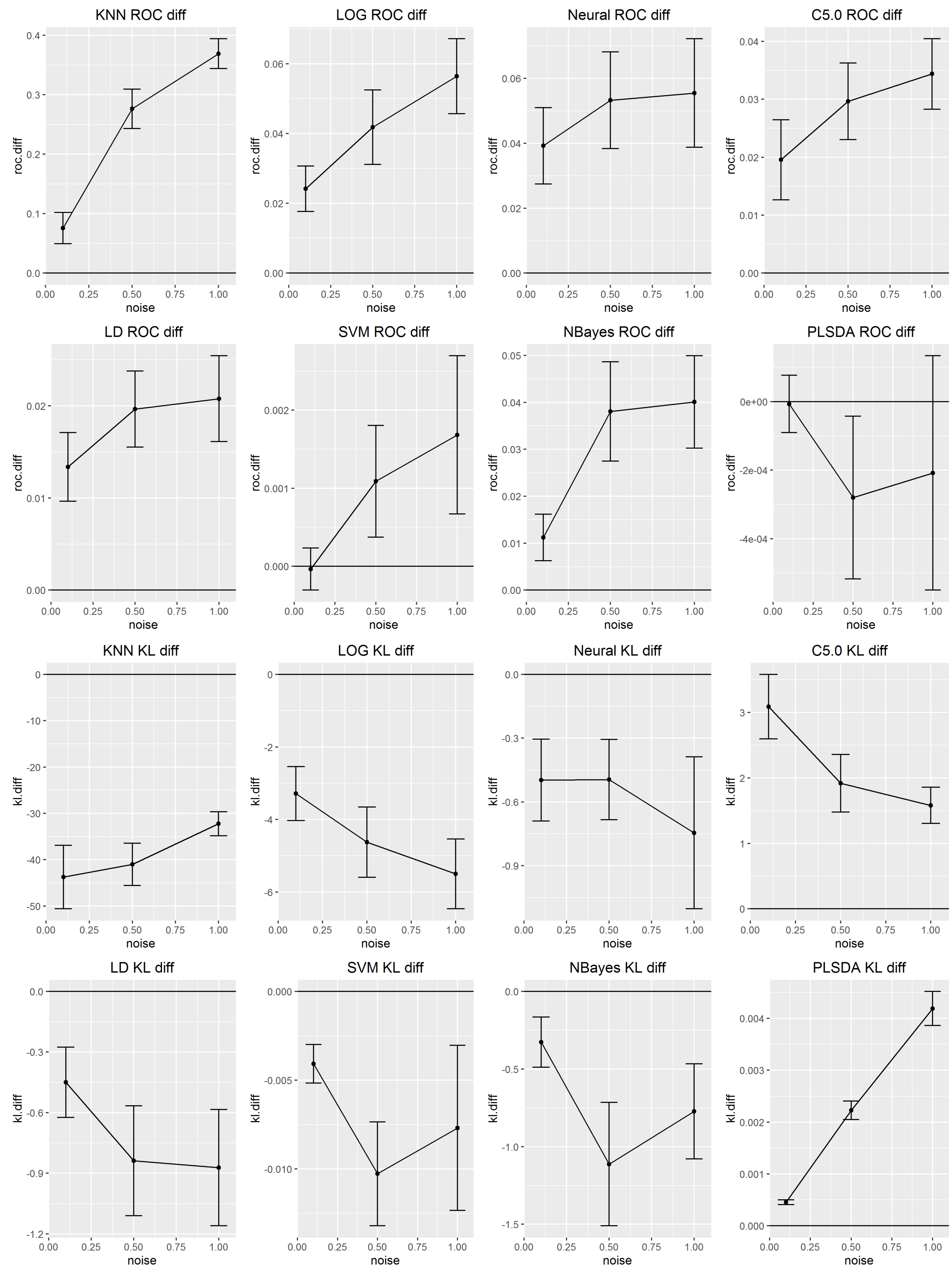


Figure 12: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 6.38

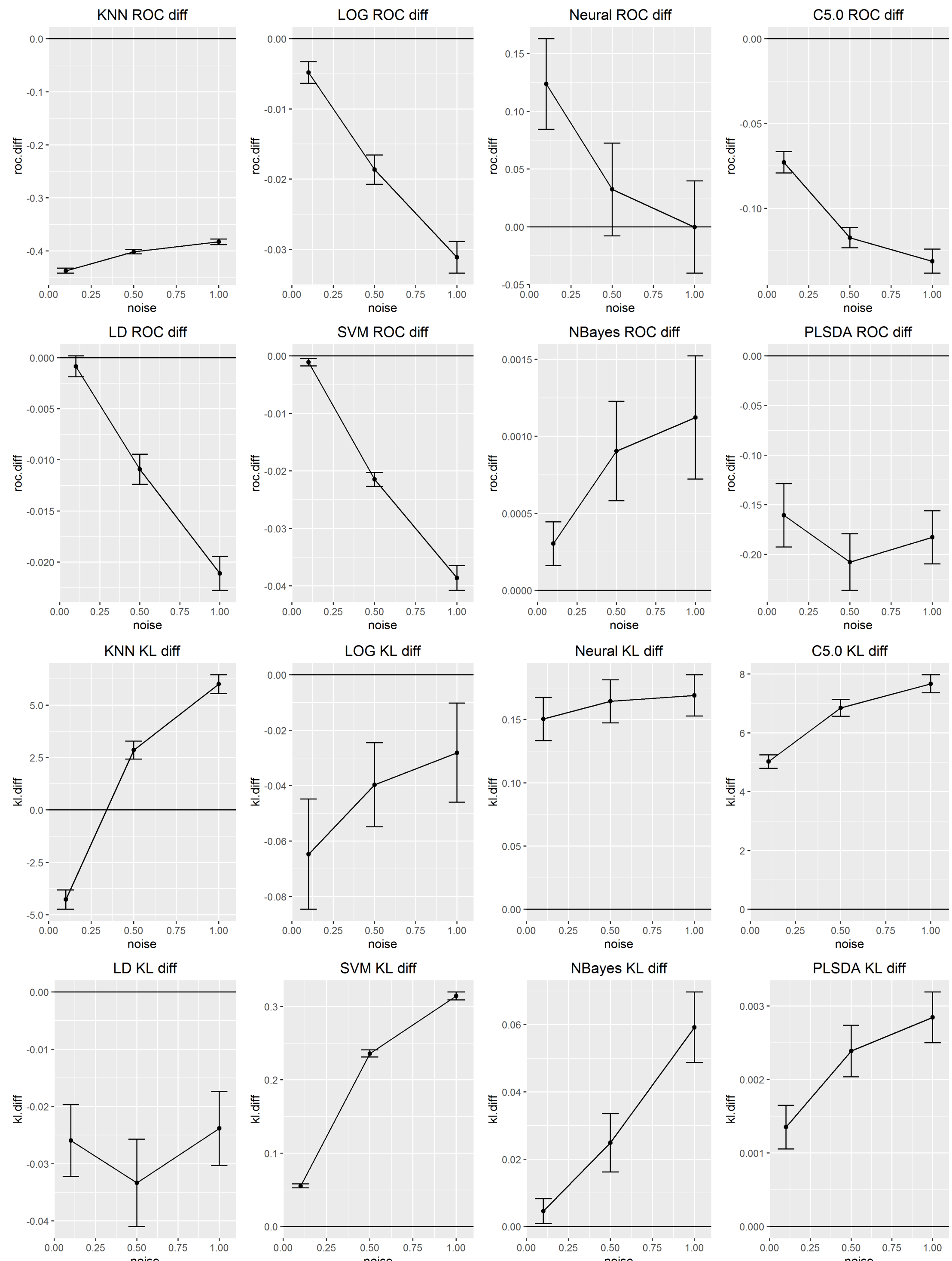


Figure 13: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 8.79

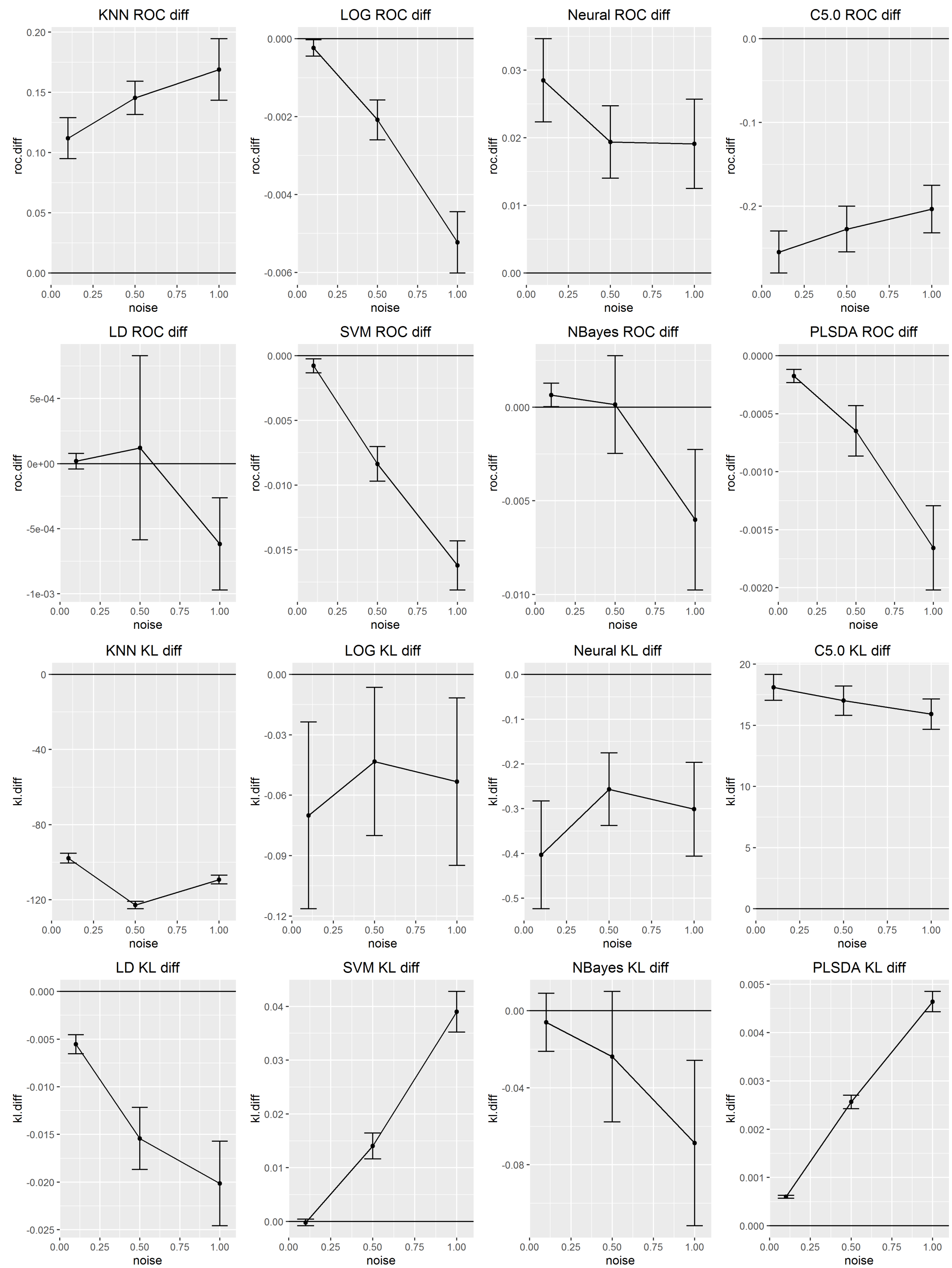


Figure 14: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 9.14

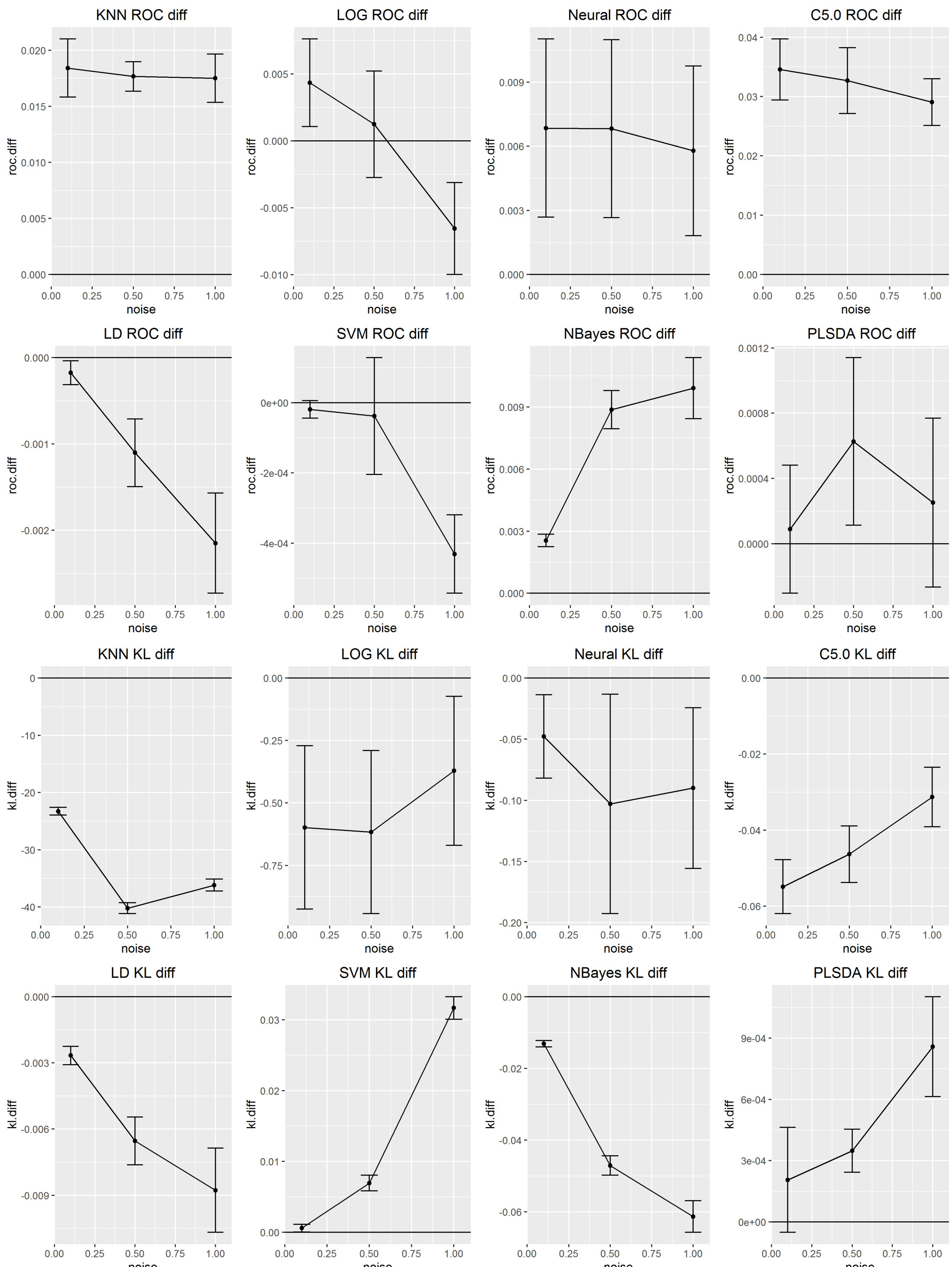


Figure 15: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 9.98

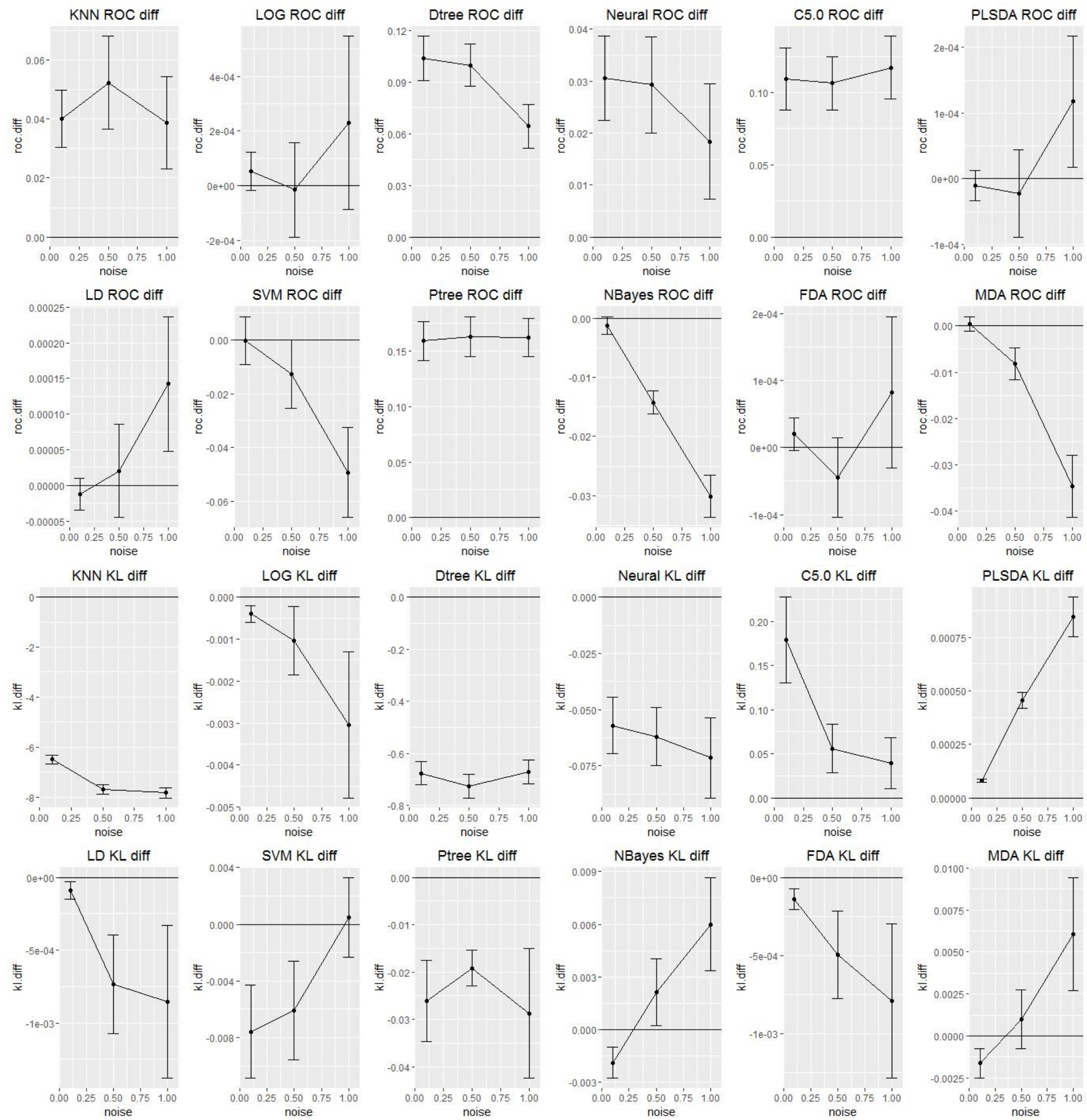


Figure 16: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 10.00

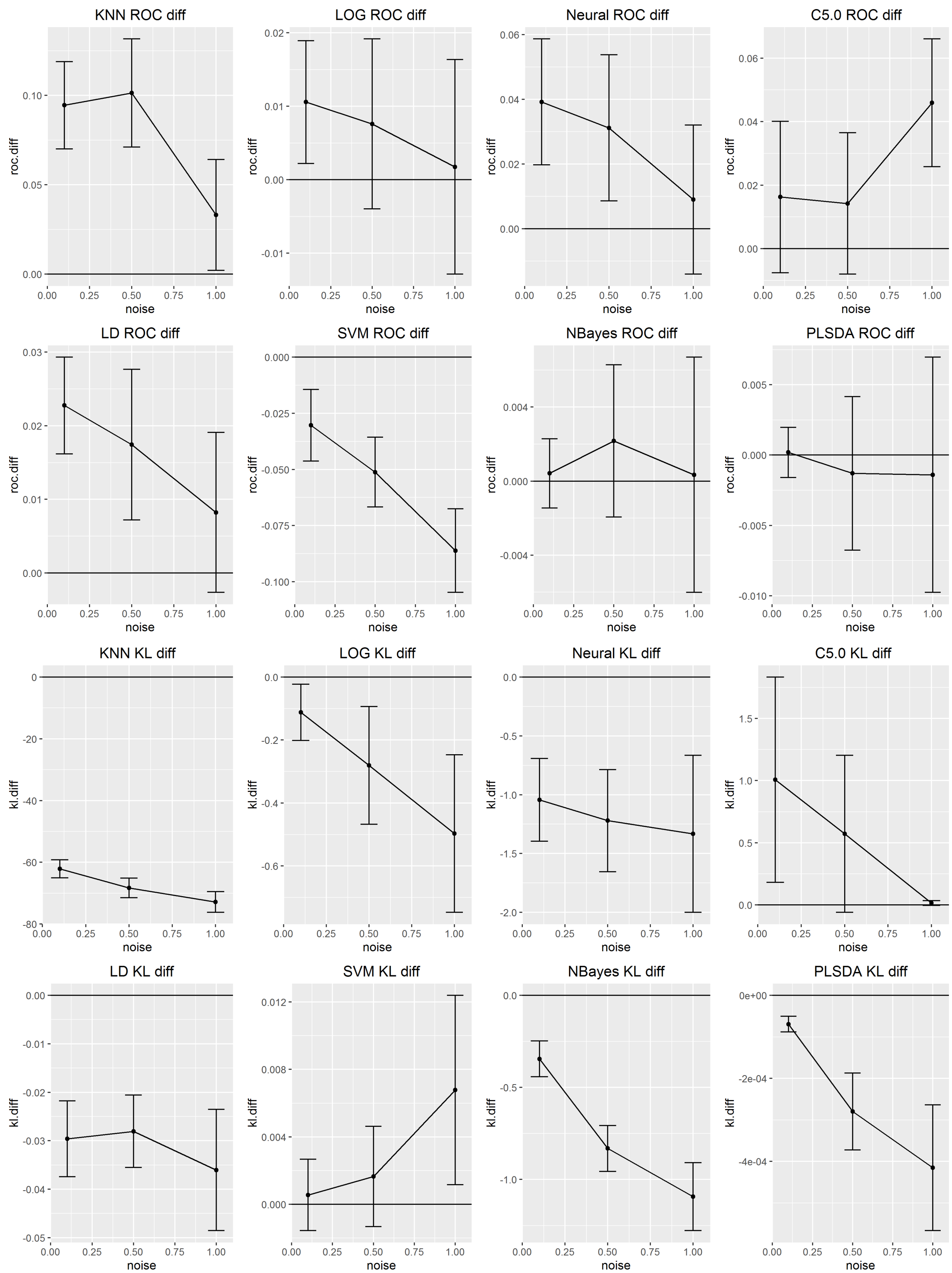


Figure 17: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 10.29

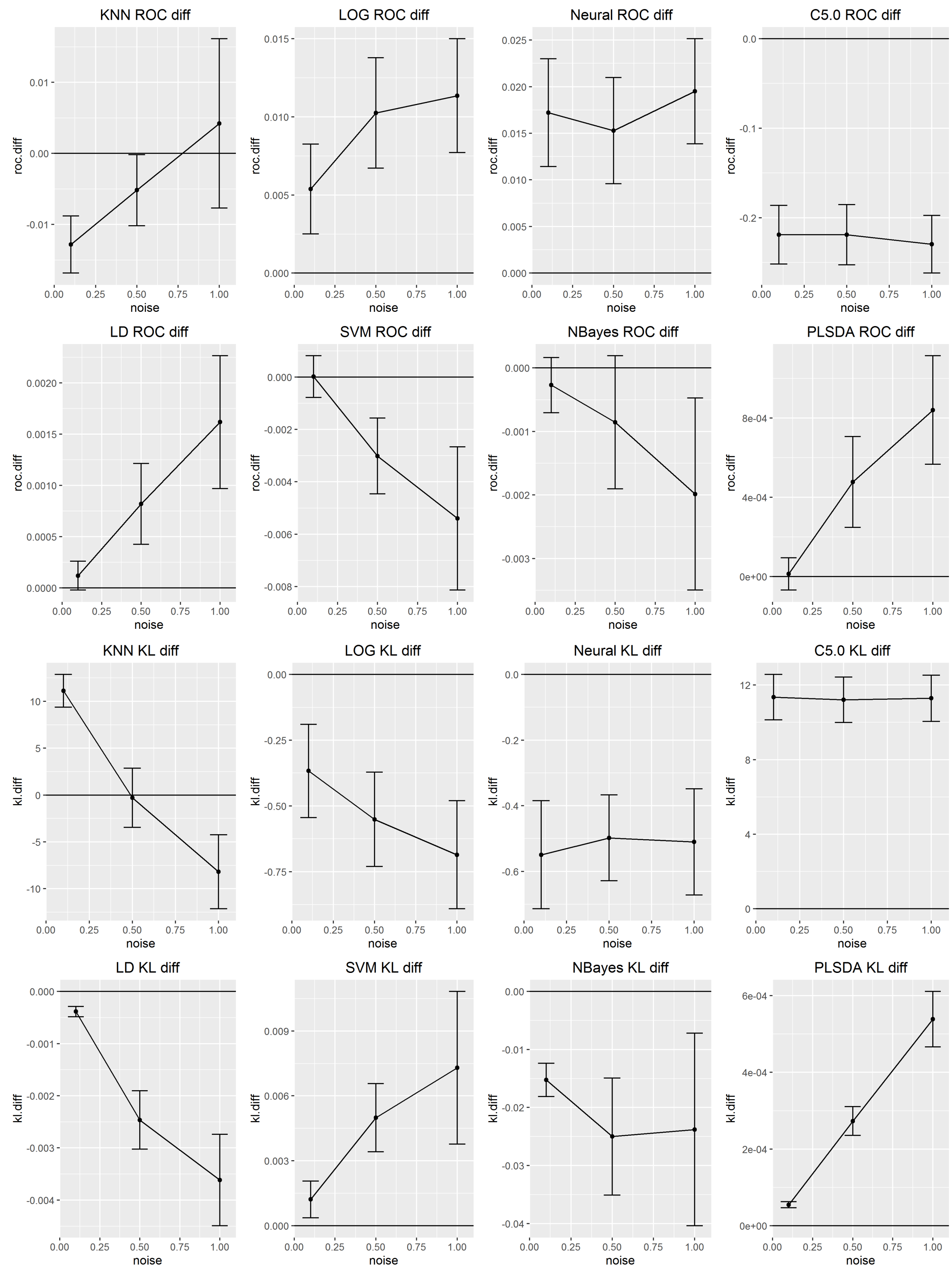


Figure 18: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 10.97

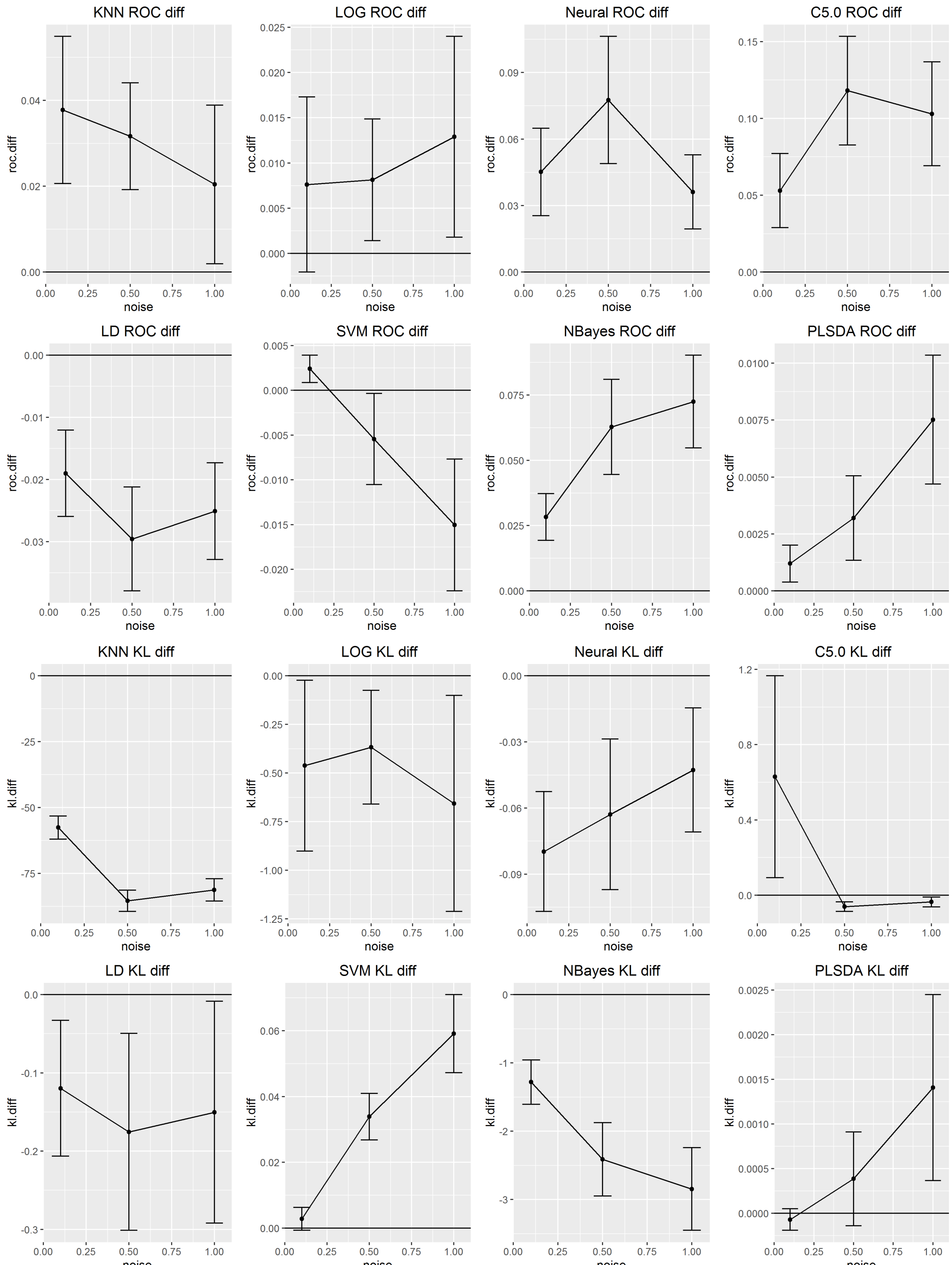


Figure 19: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 11.00

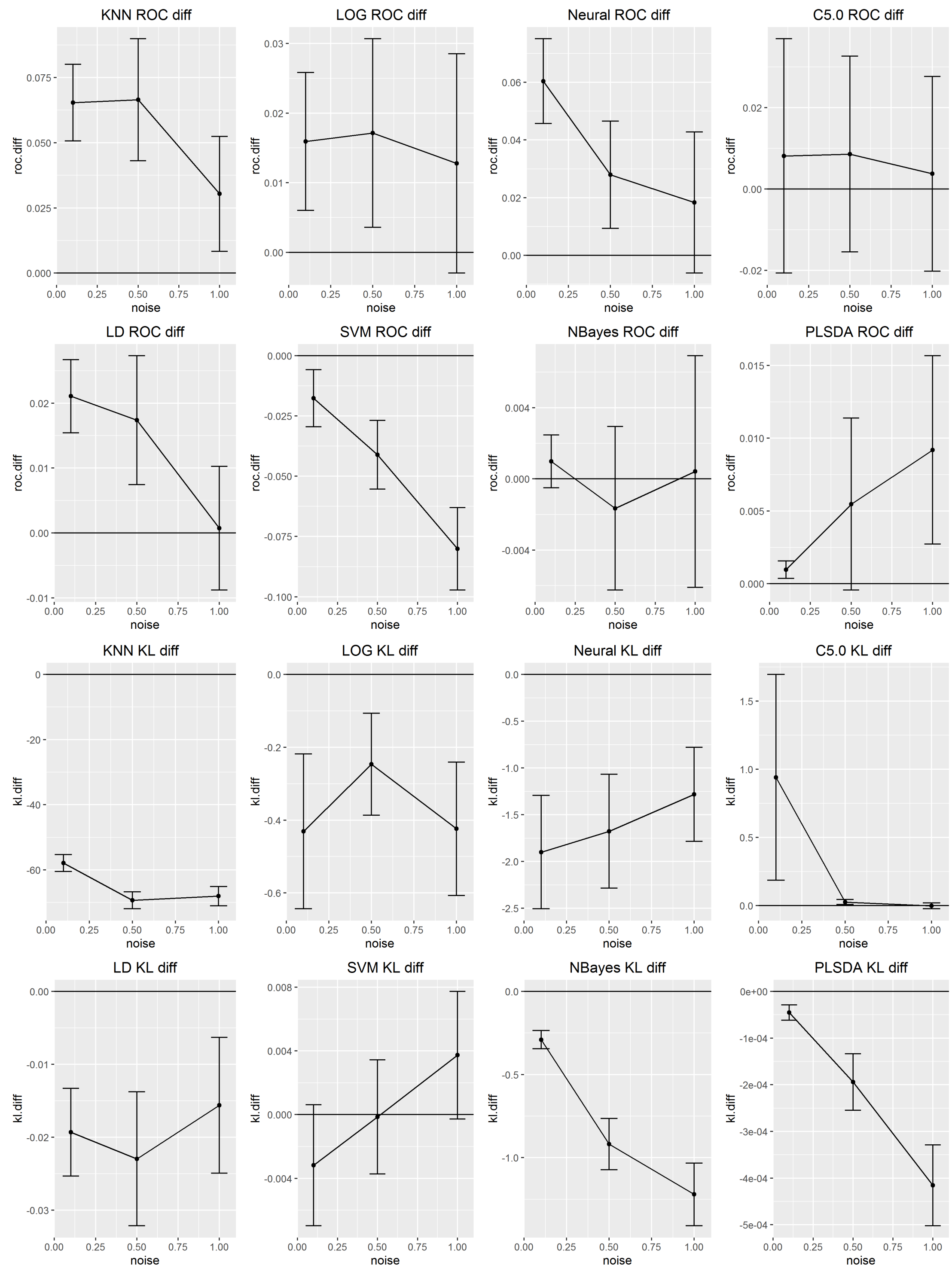


Figure 20: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 11.06

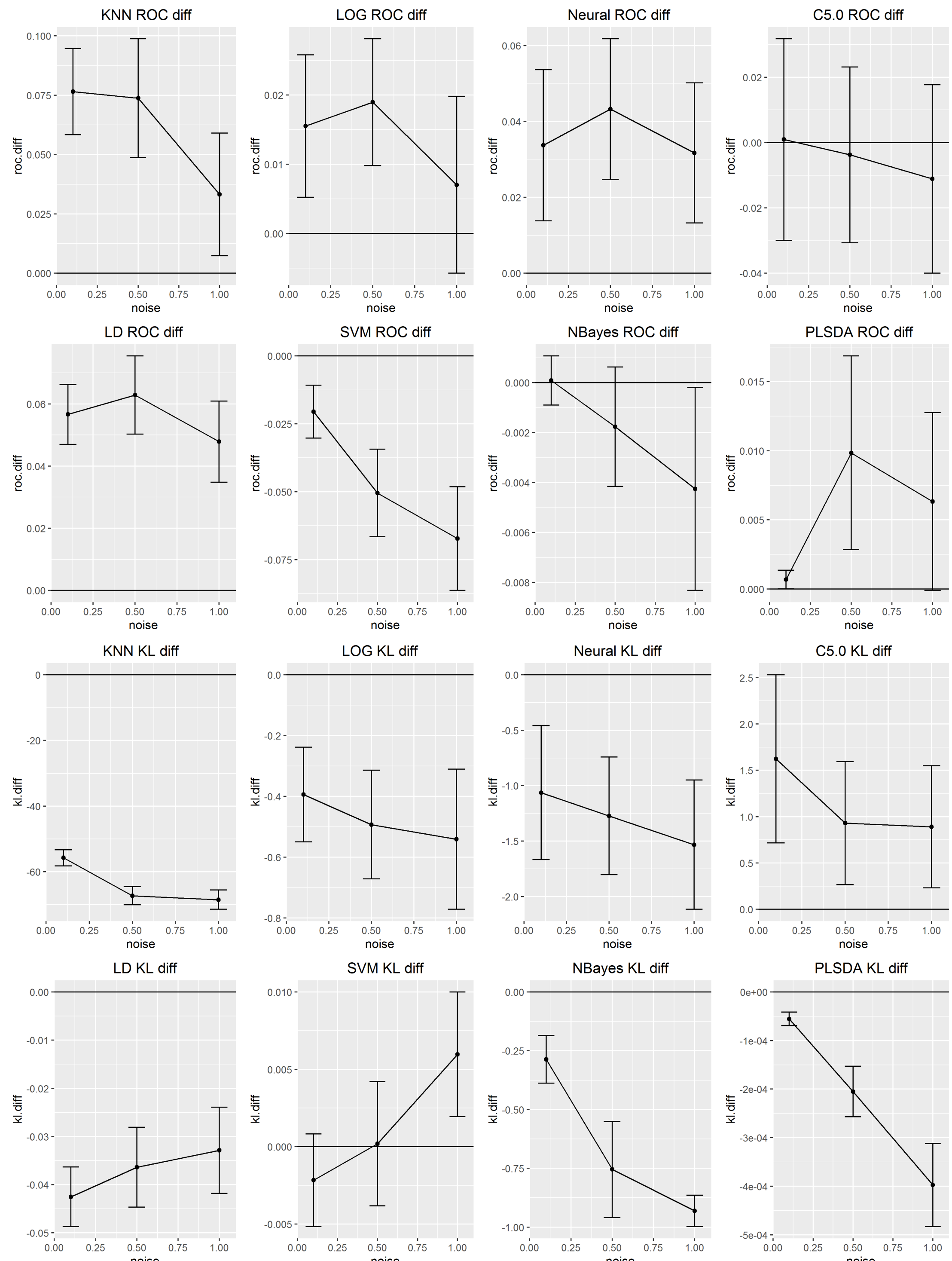


Figure 21: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 11.59

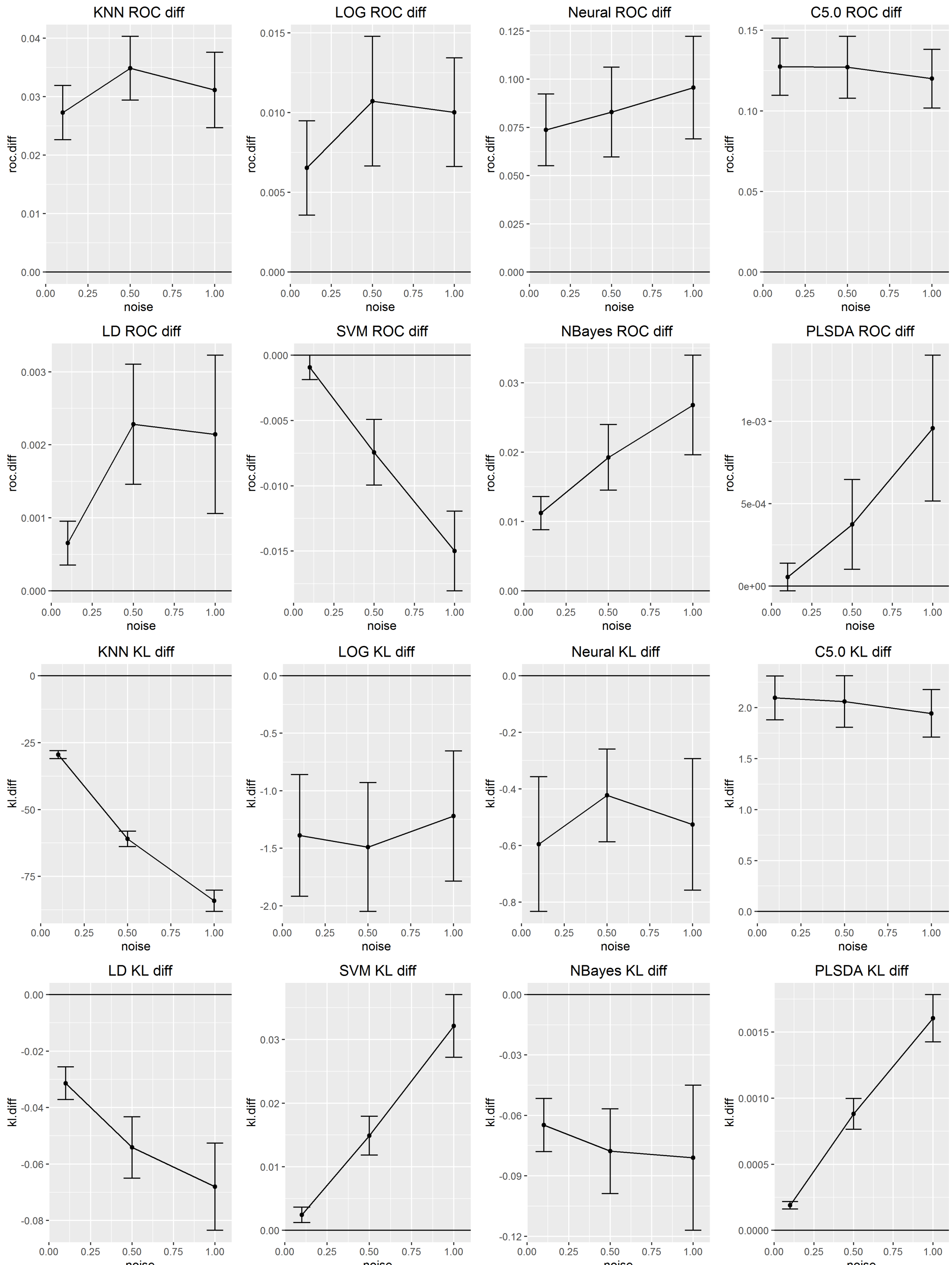


Figure 22: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 12.28

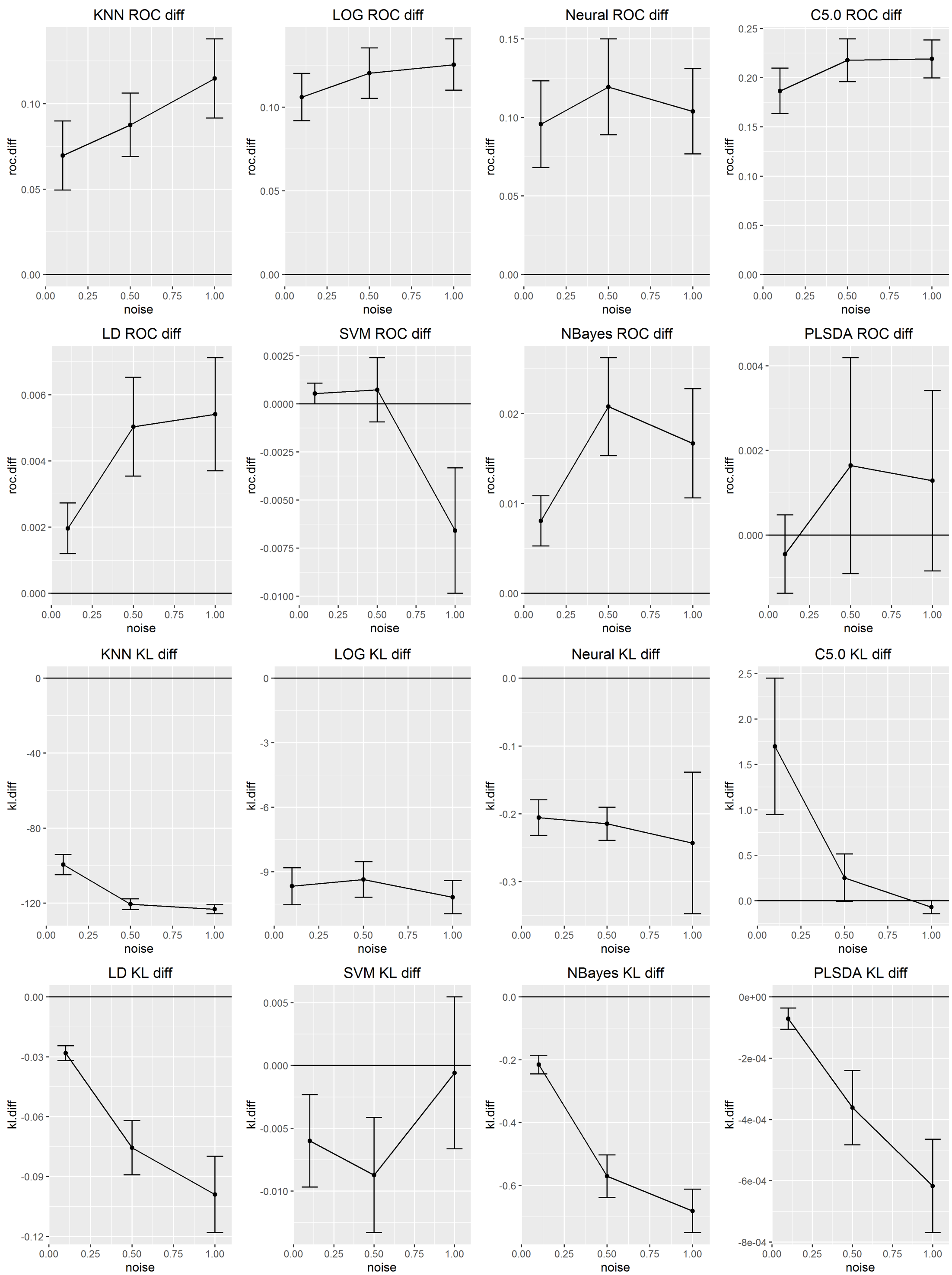


Figure 23: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 12.62

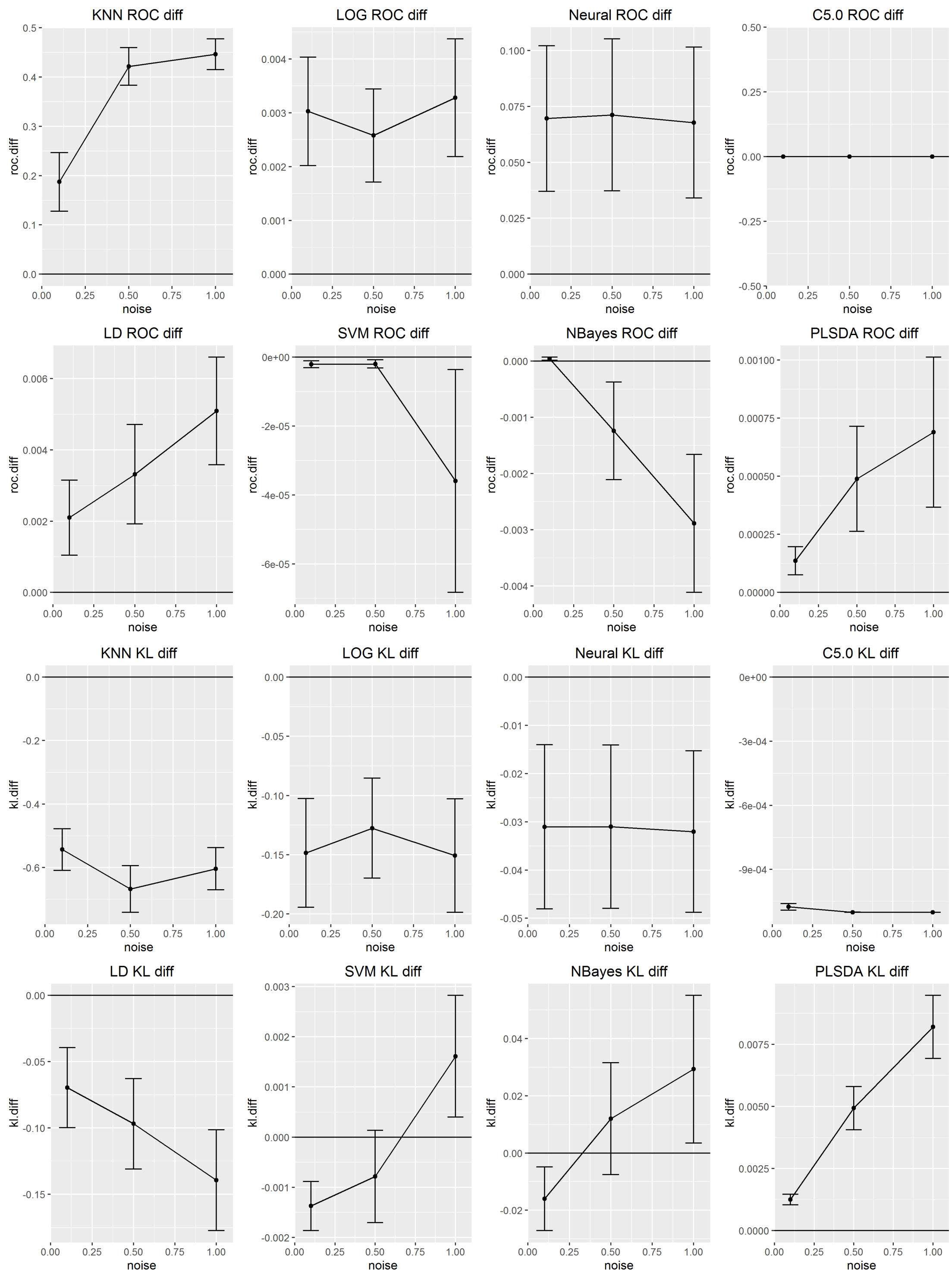


Figure 24: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 13.87

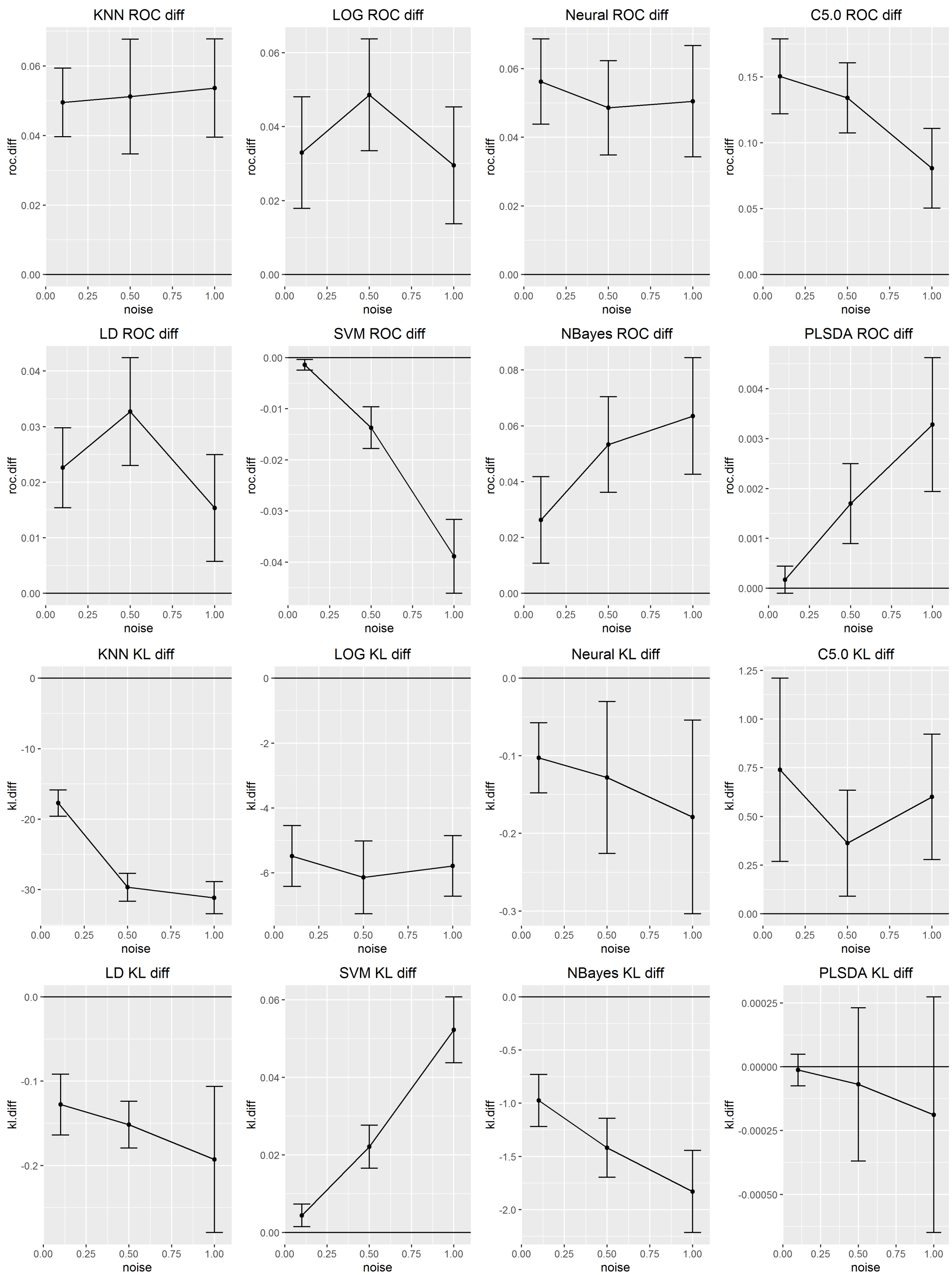


Figure 25: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 15.47

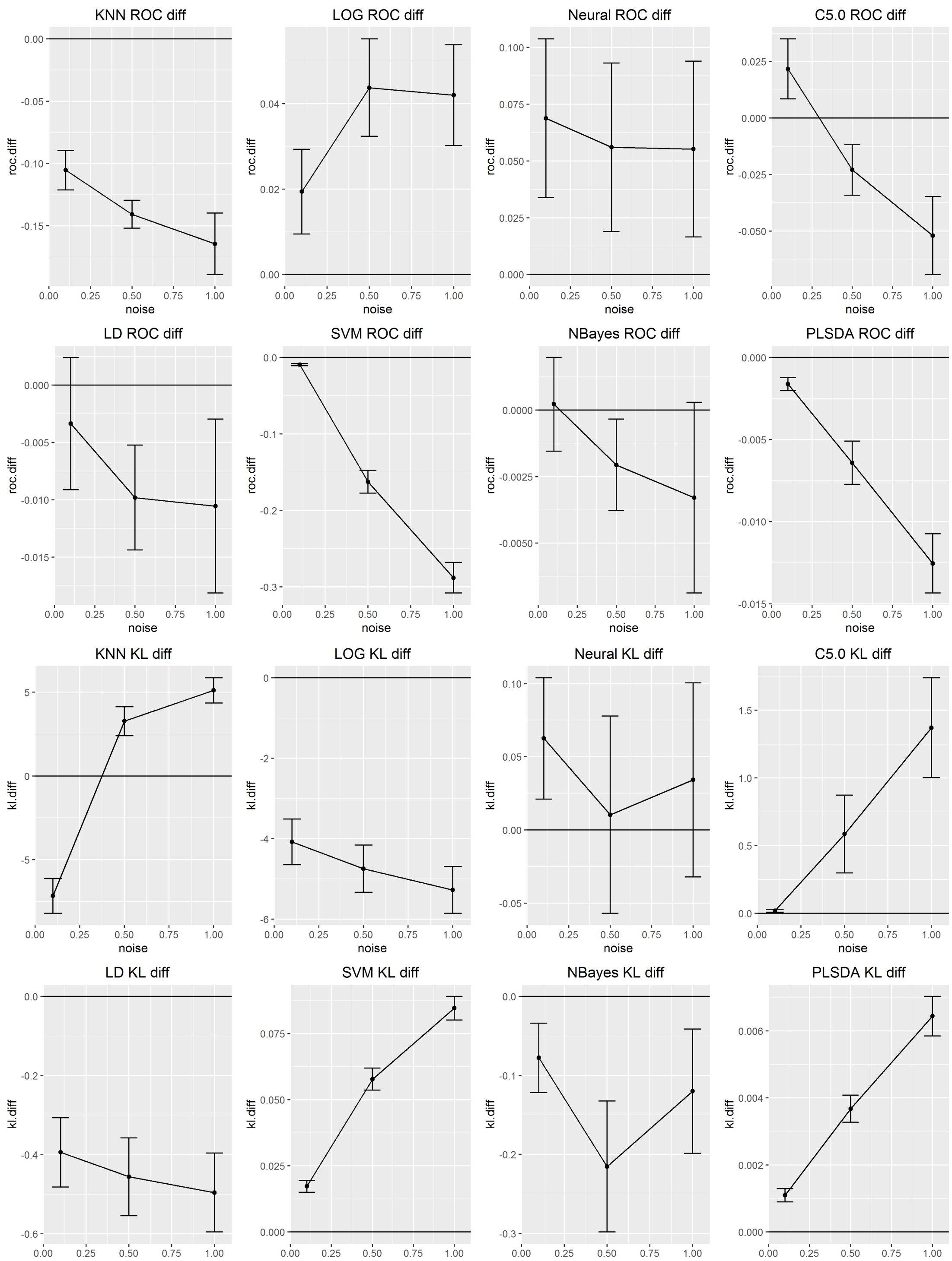


Figure 26: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 15.86

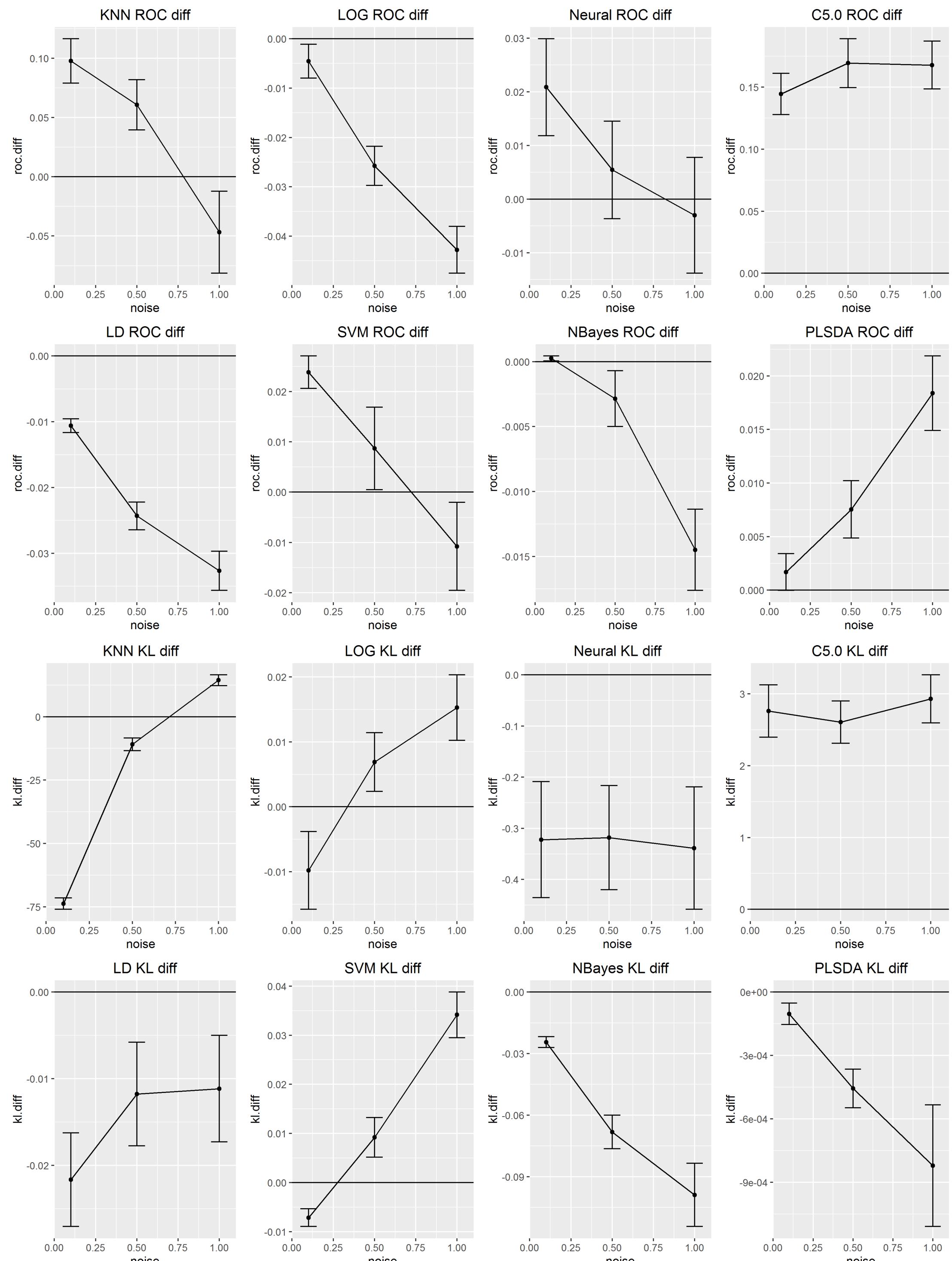


Figure 27: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 16.40

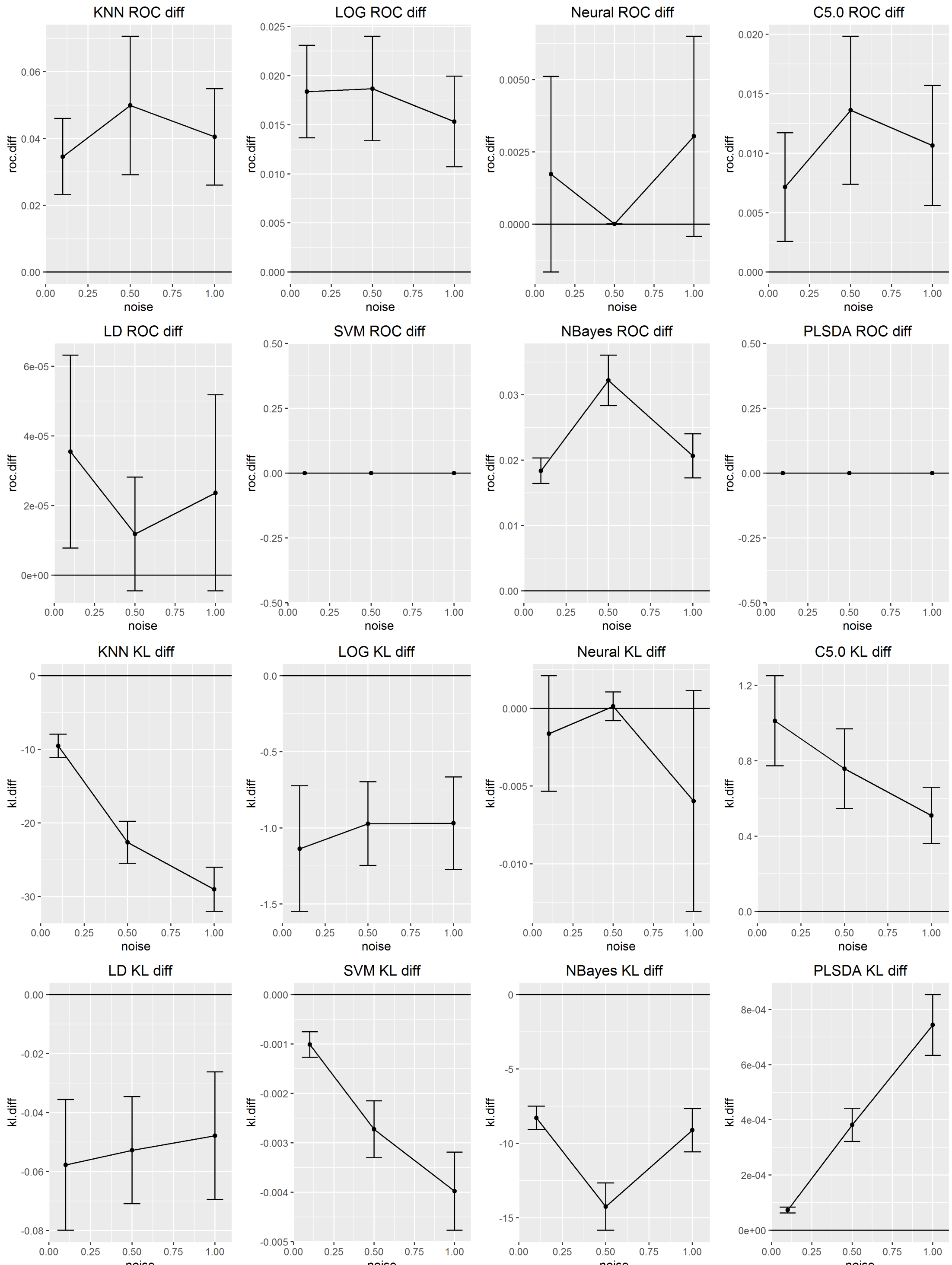


Figure 28: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 16.90

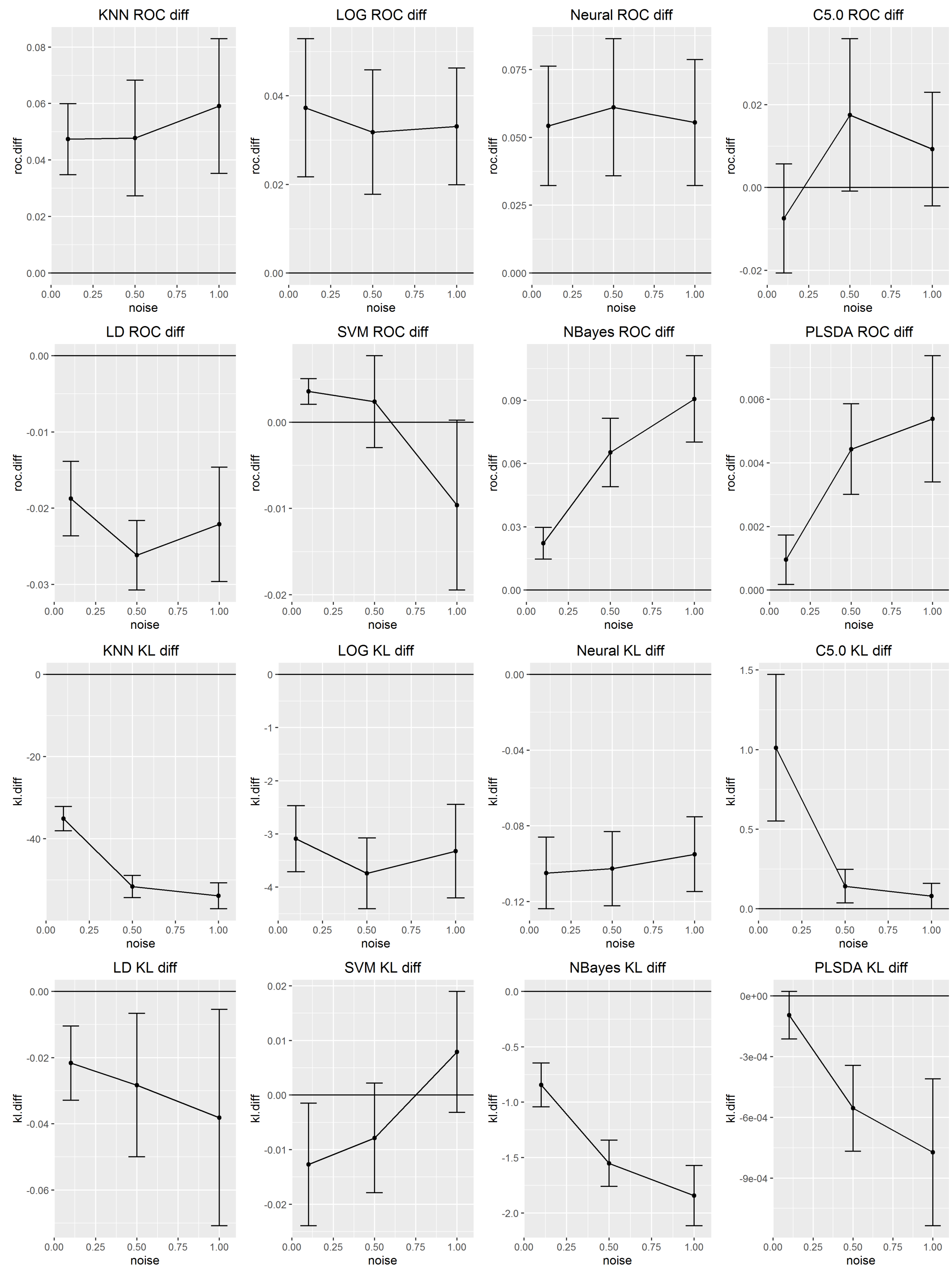


Figure 29: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 19.44

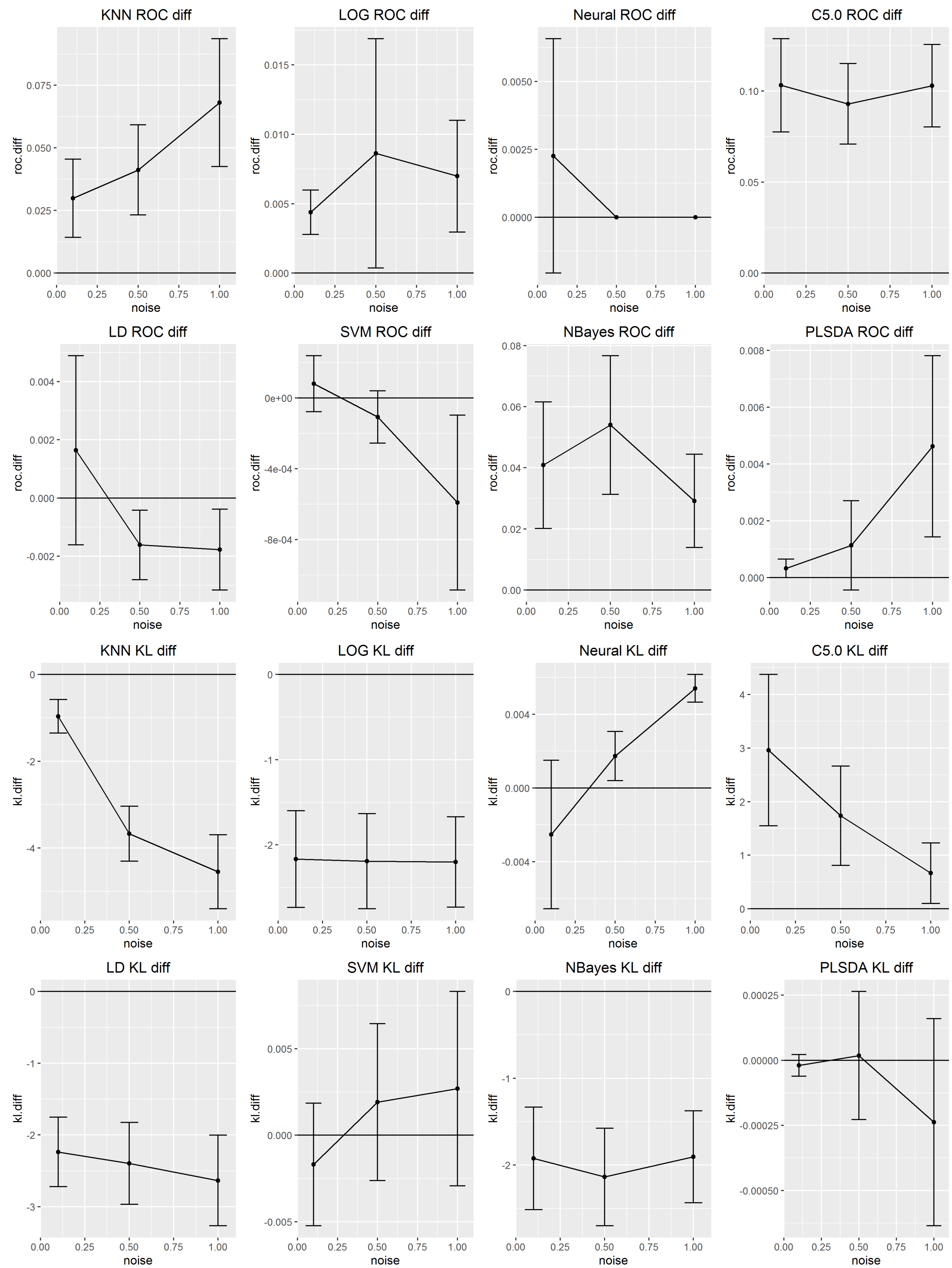


Figure 30: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 20.50

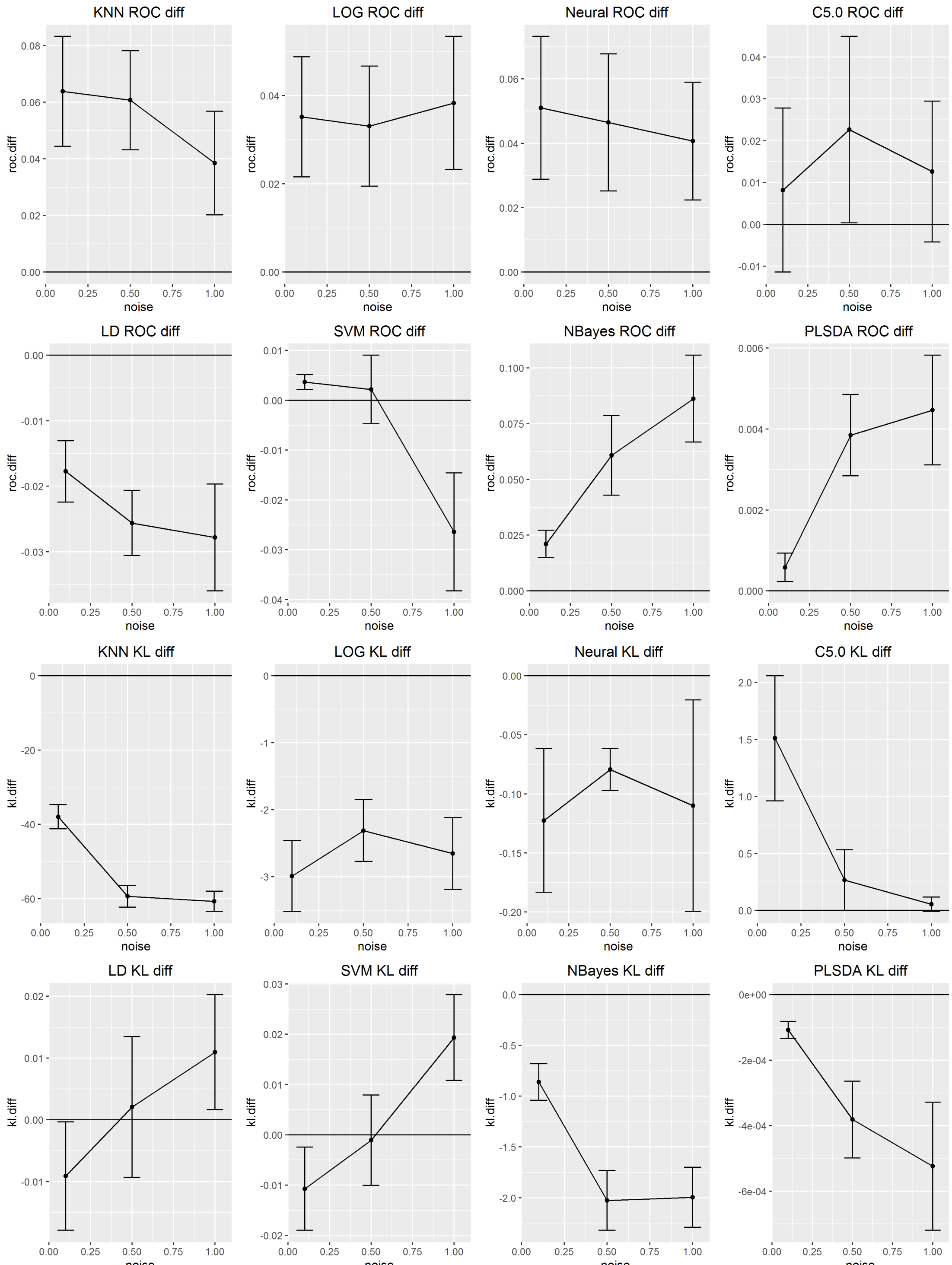


Figure 31: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 22.78

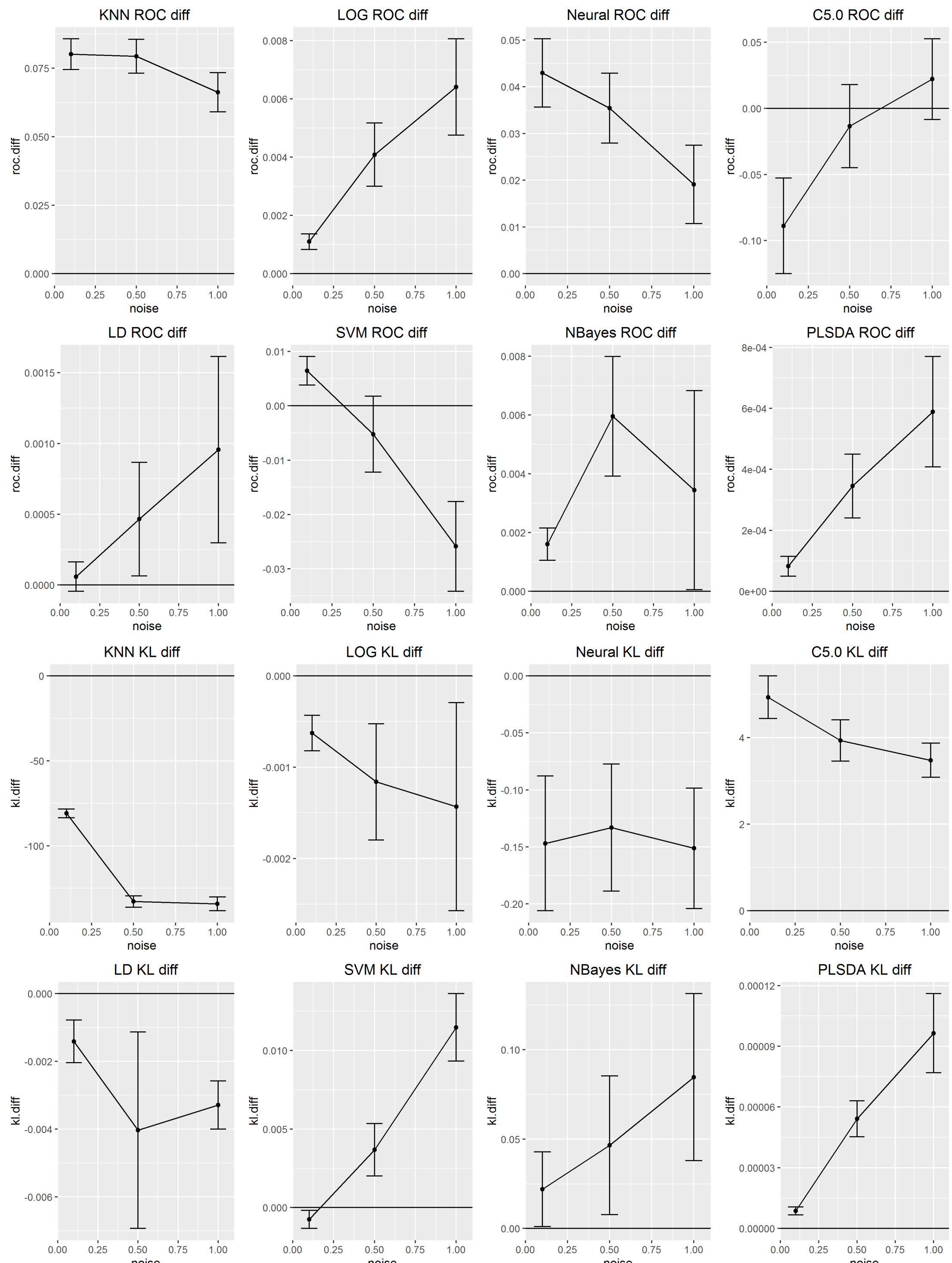


Figure 32: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 28.10

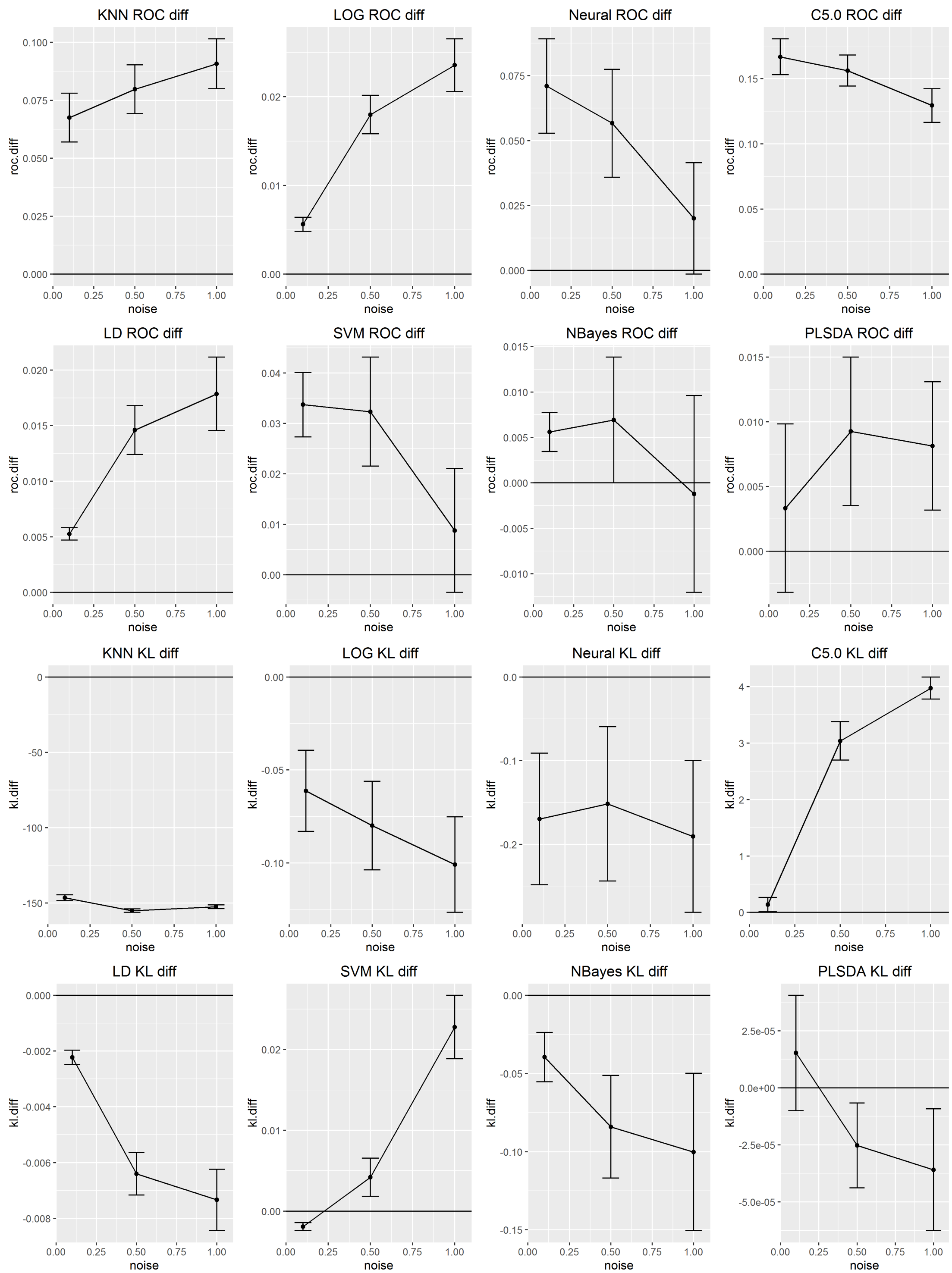


Figure 33: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 29.17

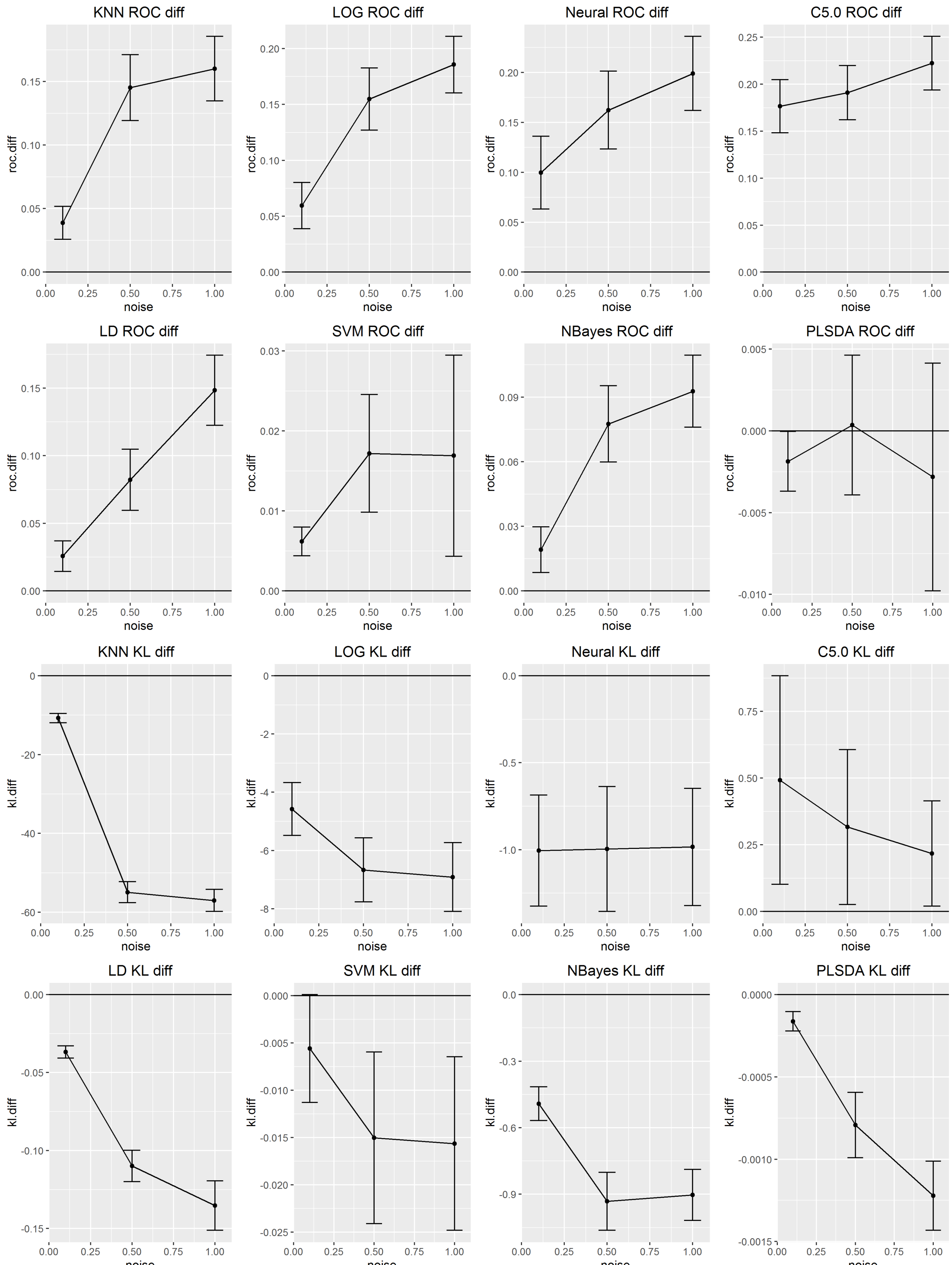


Figure 34: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 29.50

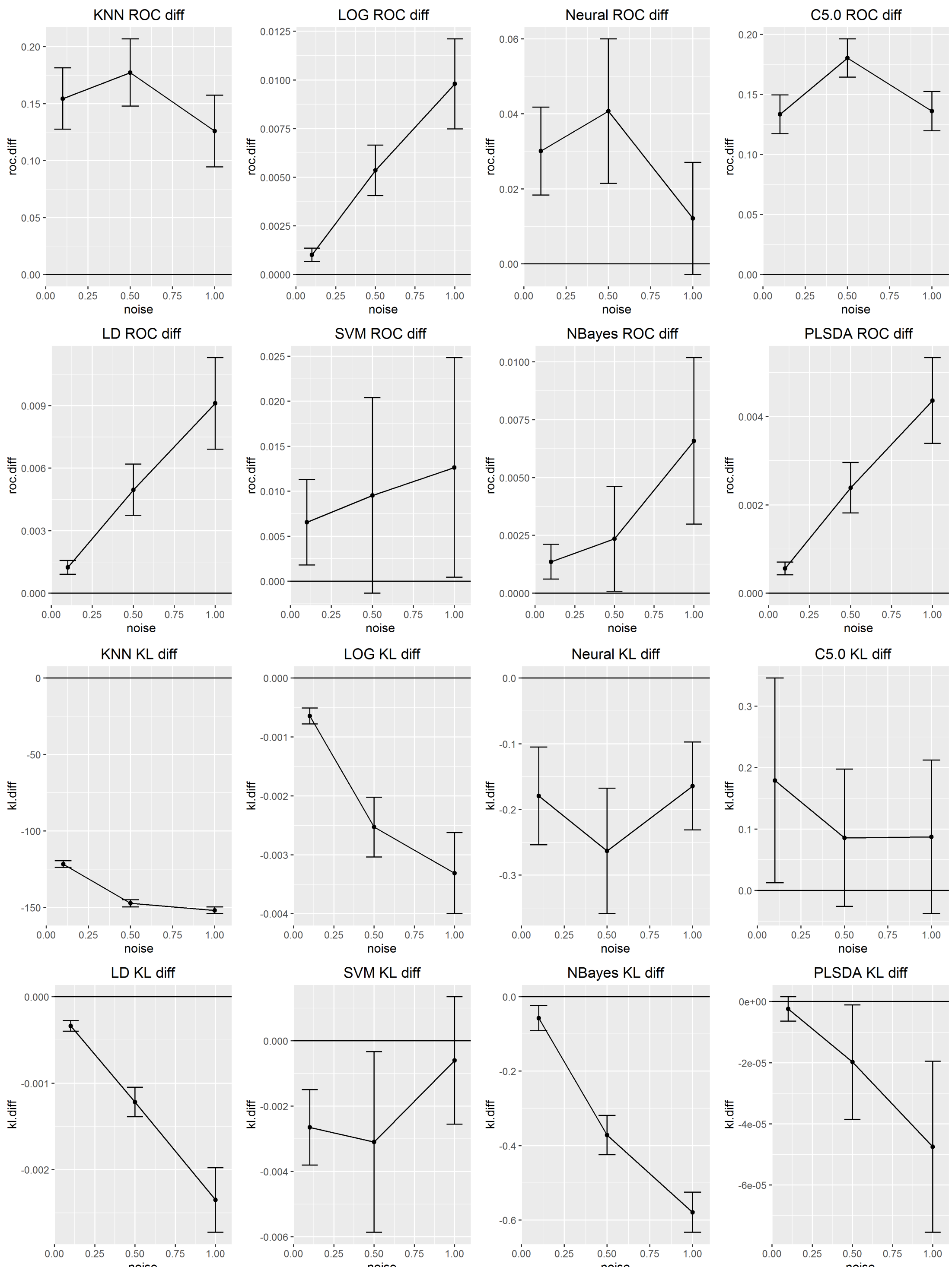


Figure 35: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 30.57

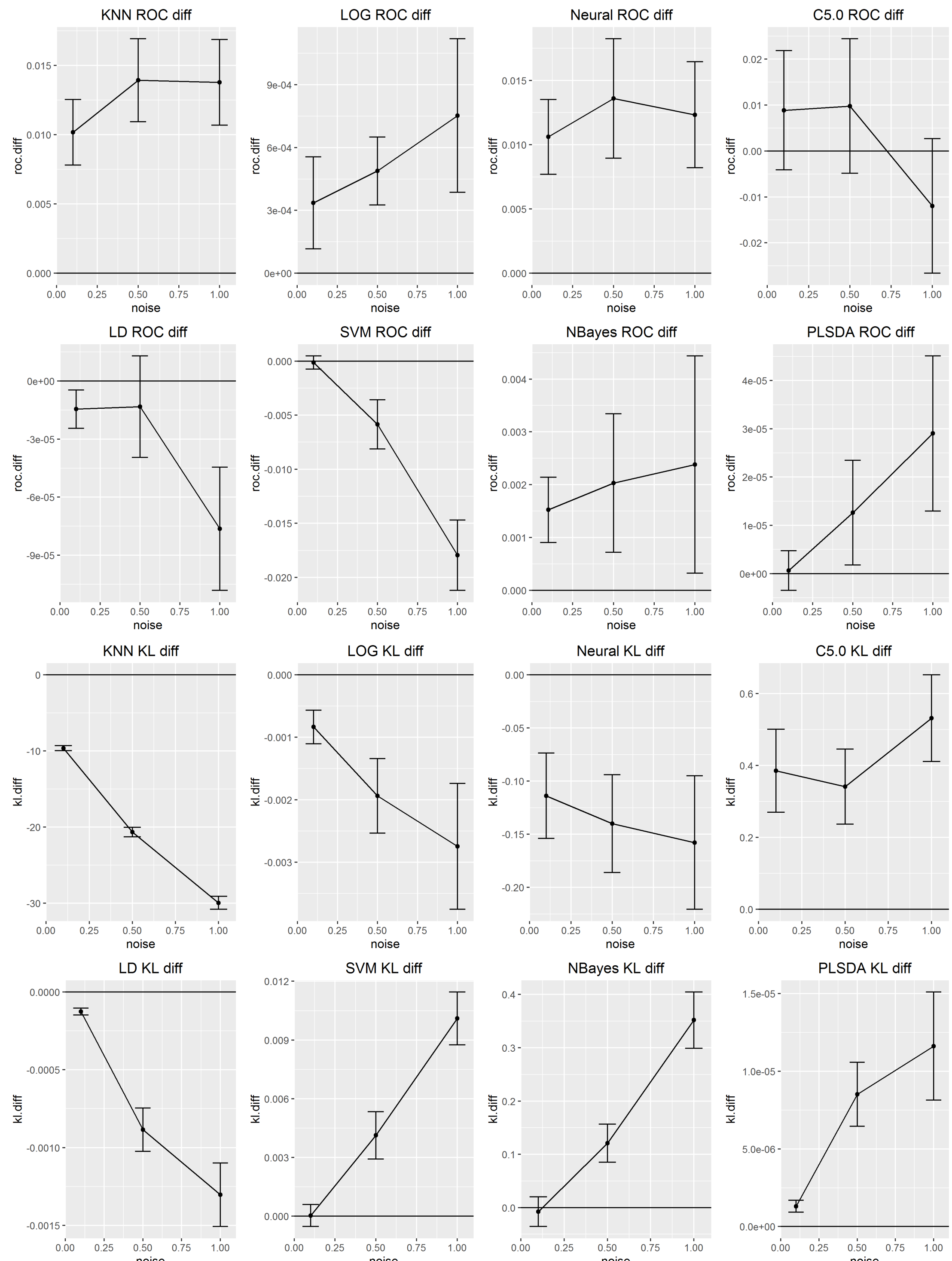


Figure 36: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 32.73

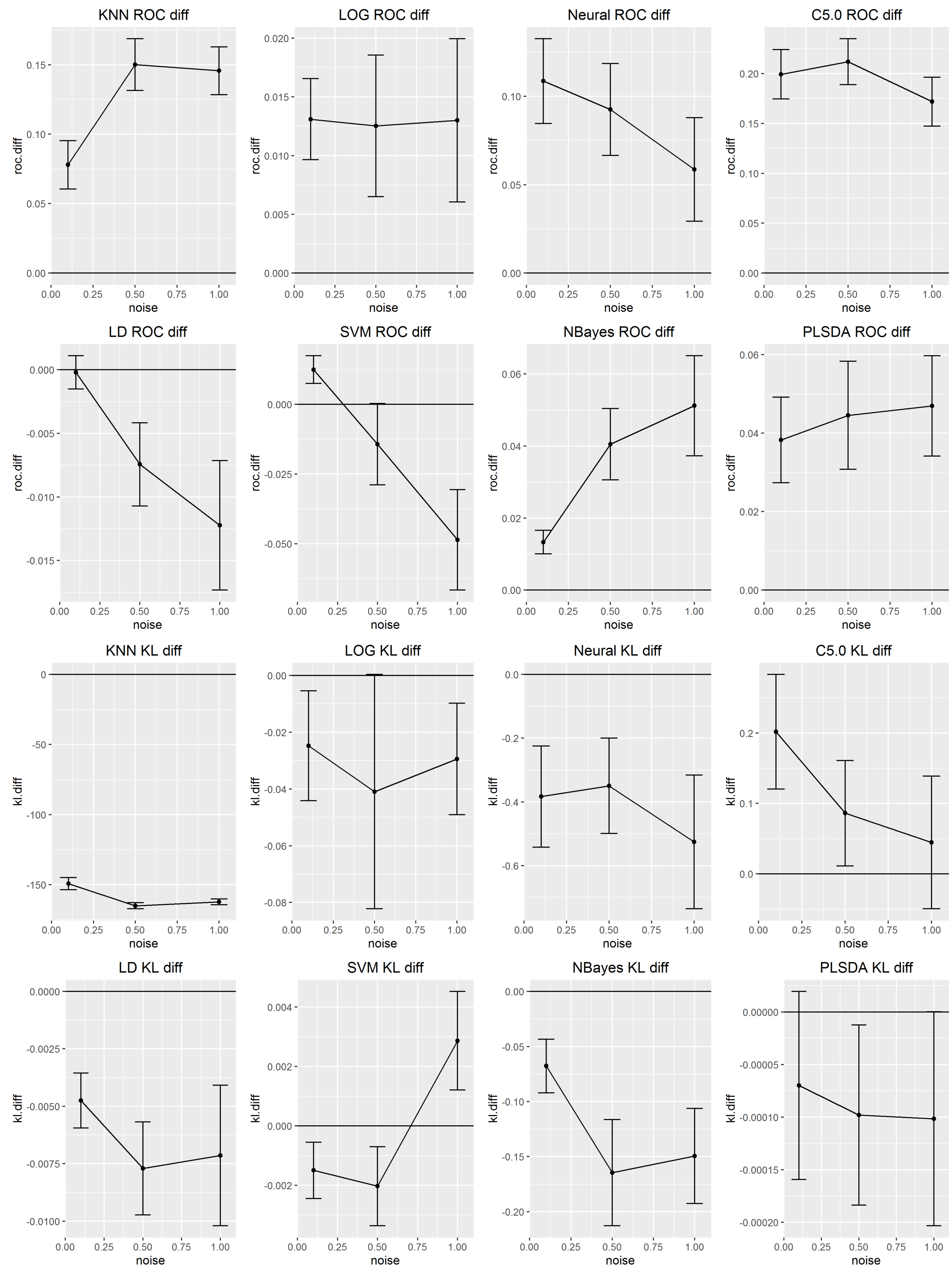


Figure 37: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 35.44

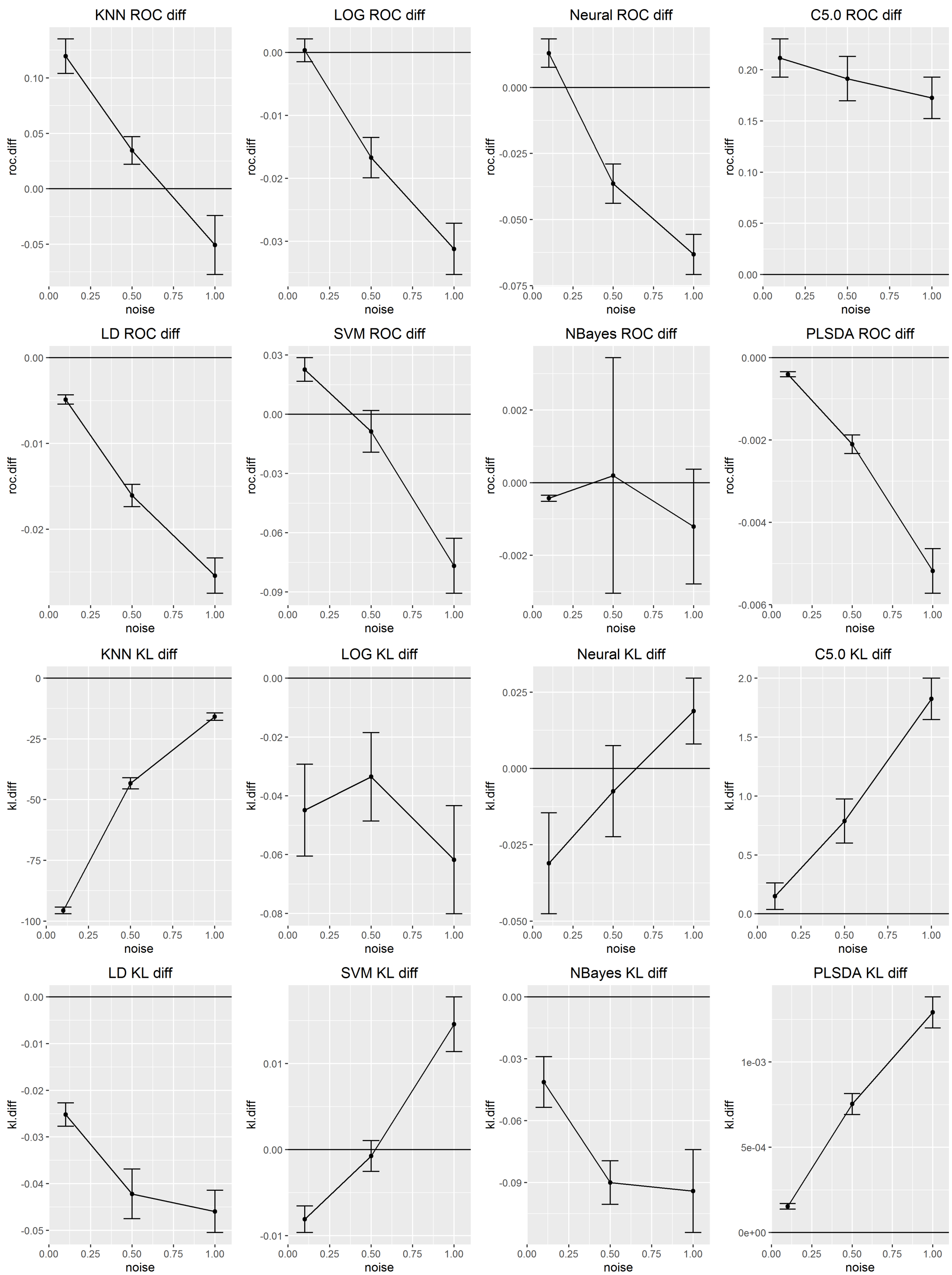


Figure 38: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 39.31

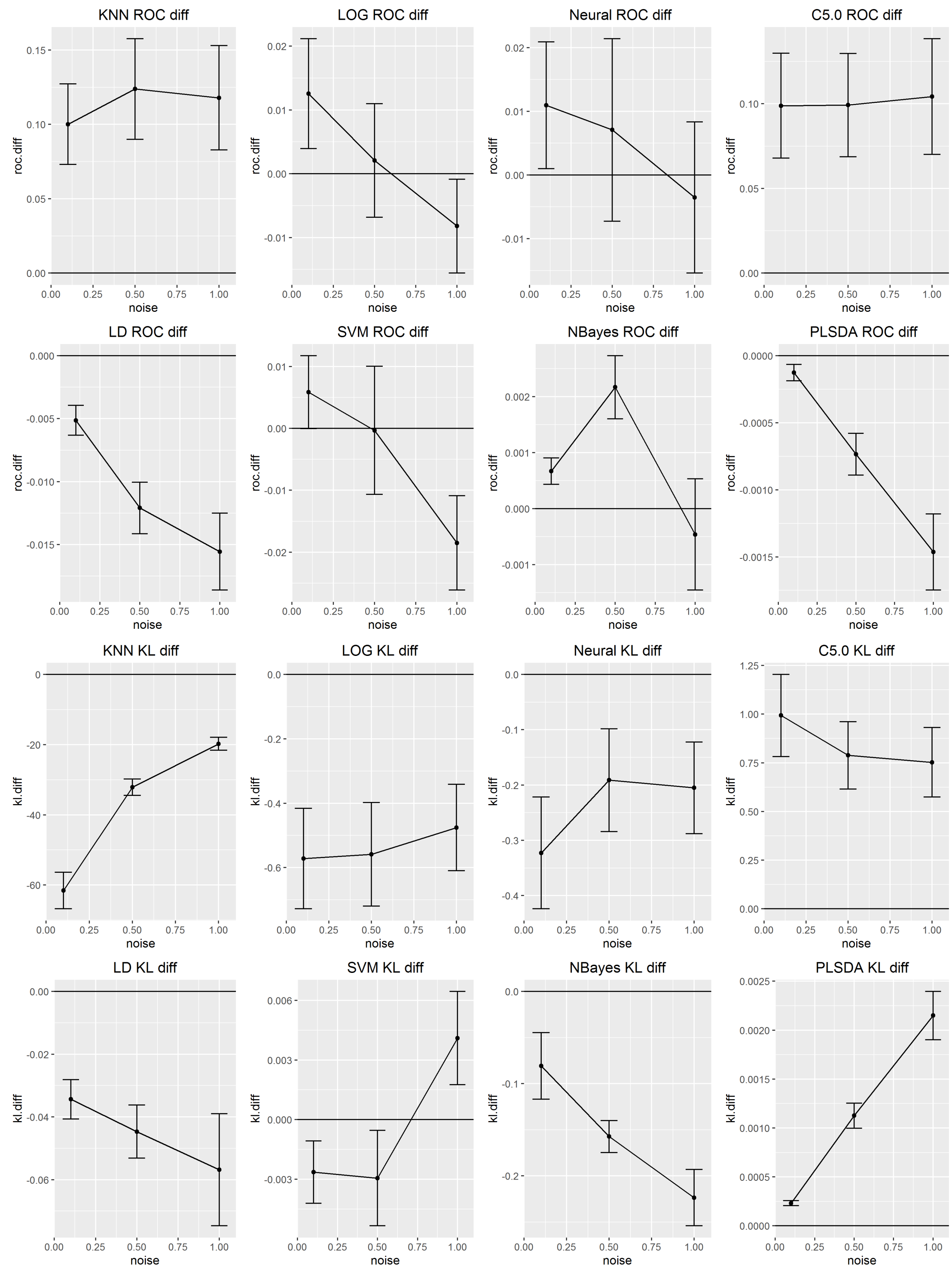


Figure 39: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 40.50

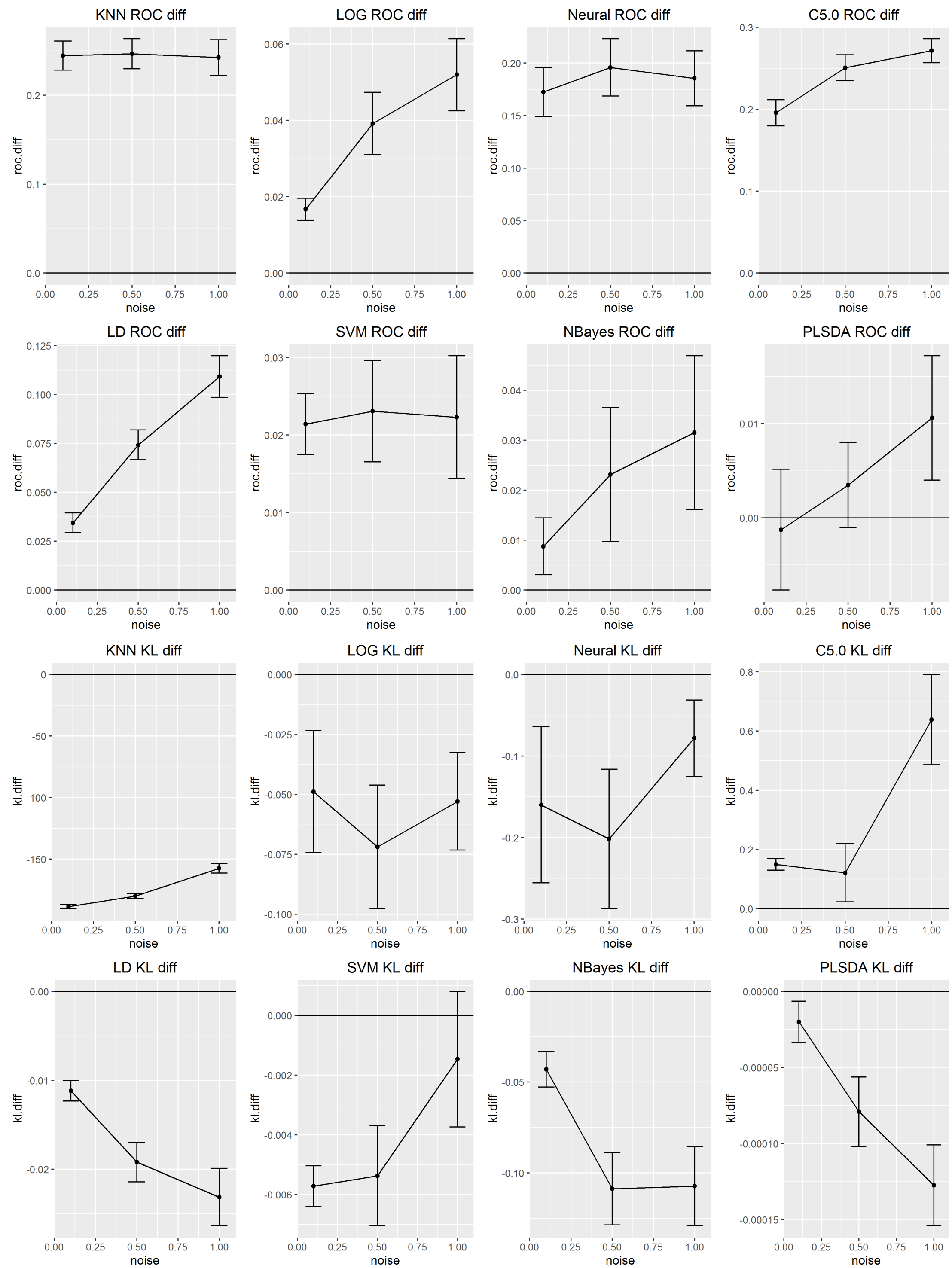


Figure 40: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 58.28

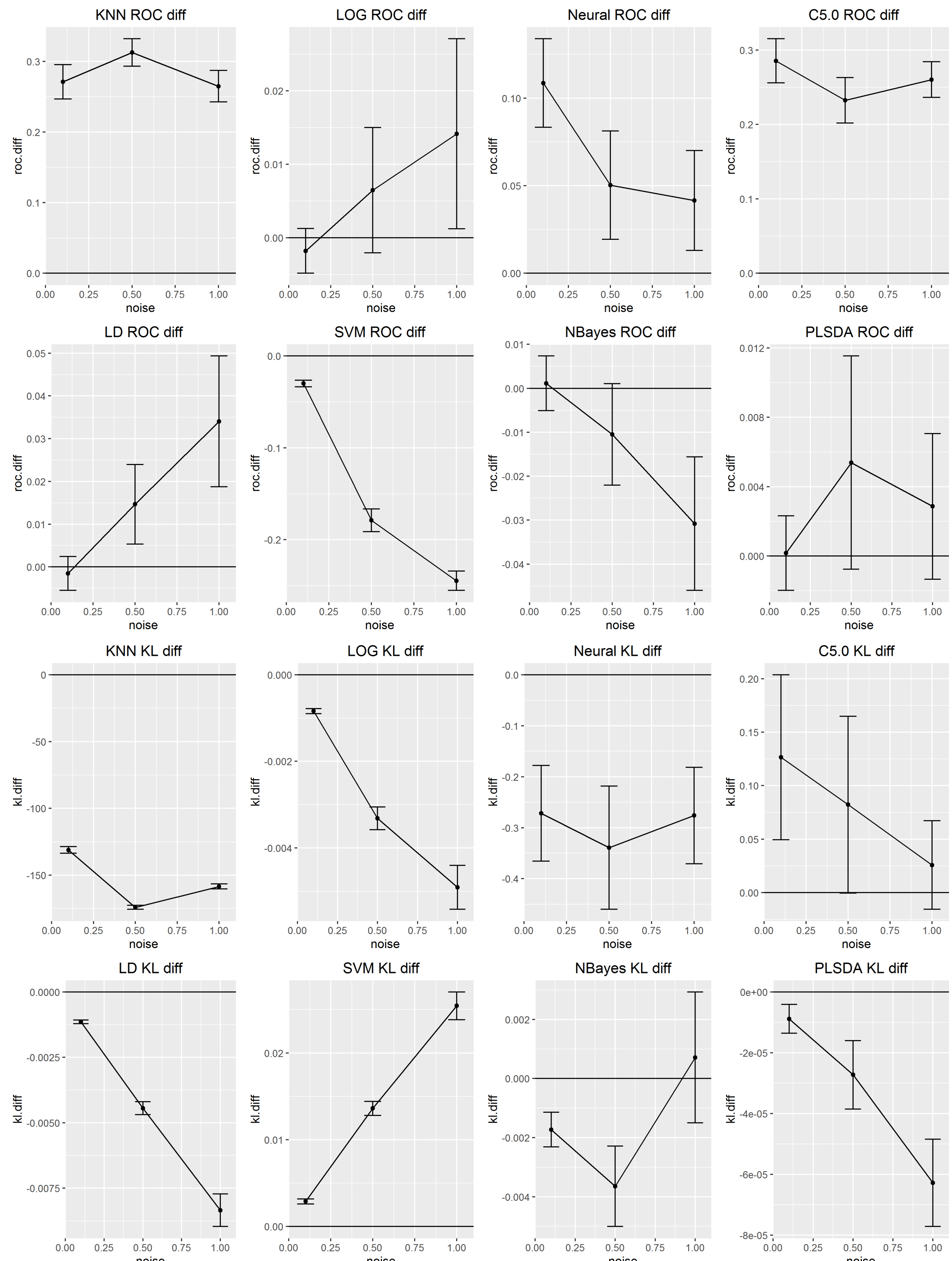


Figure 41: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 58.40

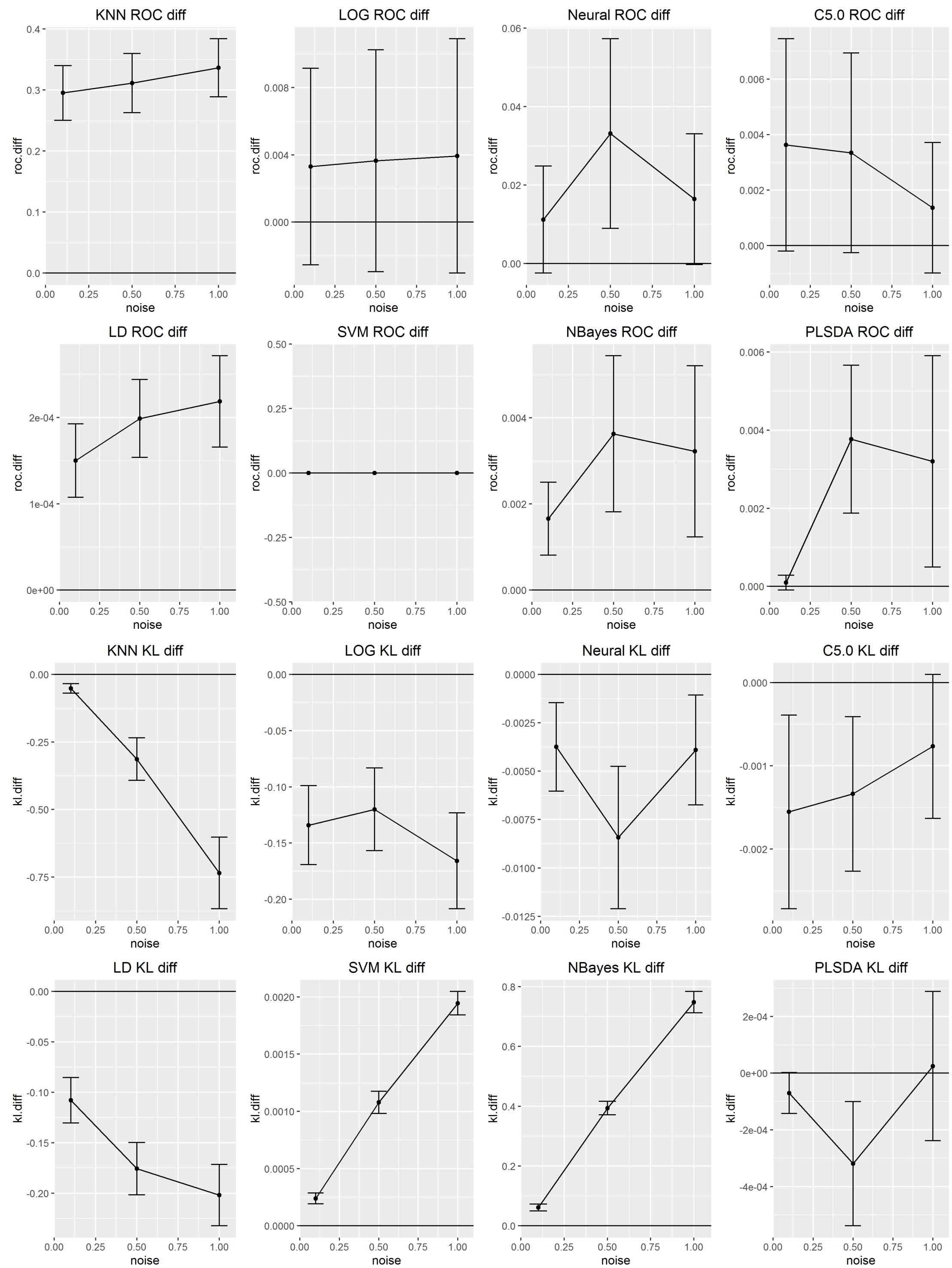


Figure 42: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 66.67

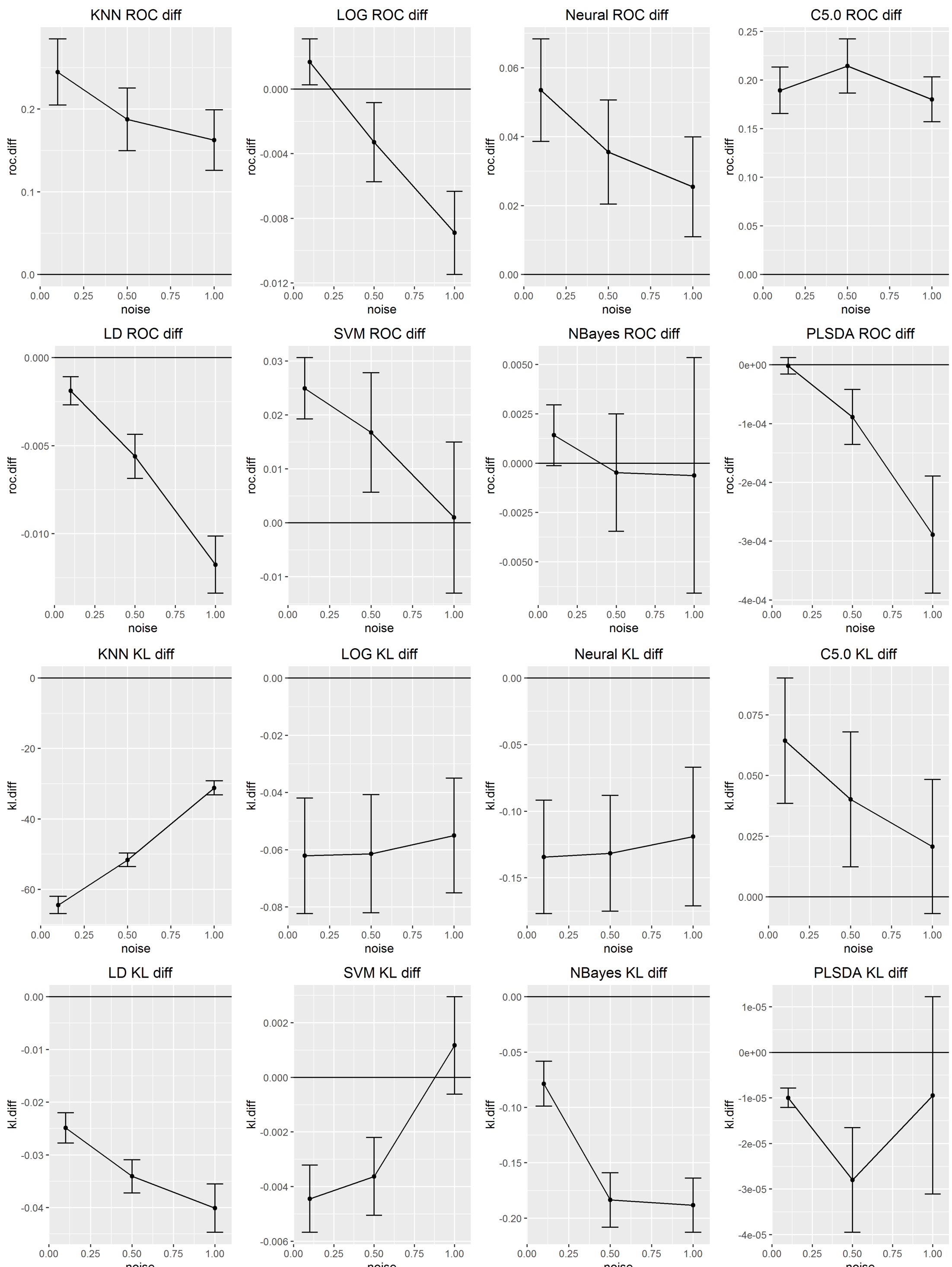


Figure 43: △ROC (top) and △KL-distance (bottom) outcomes for the binary data set with IR = 77.69

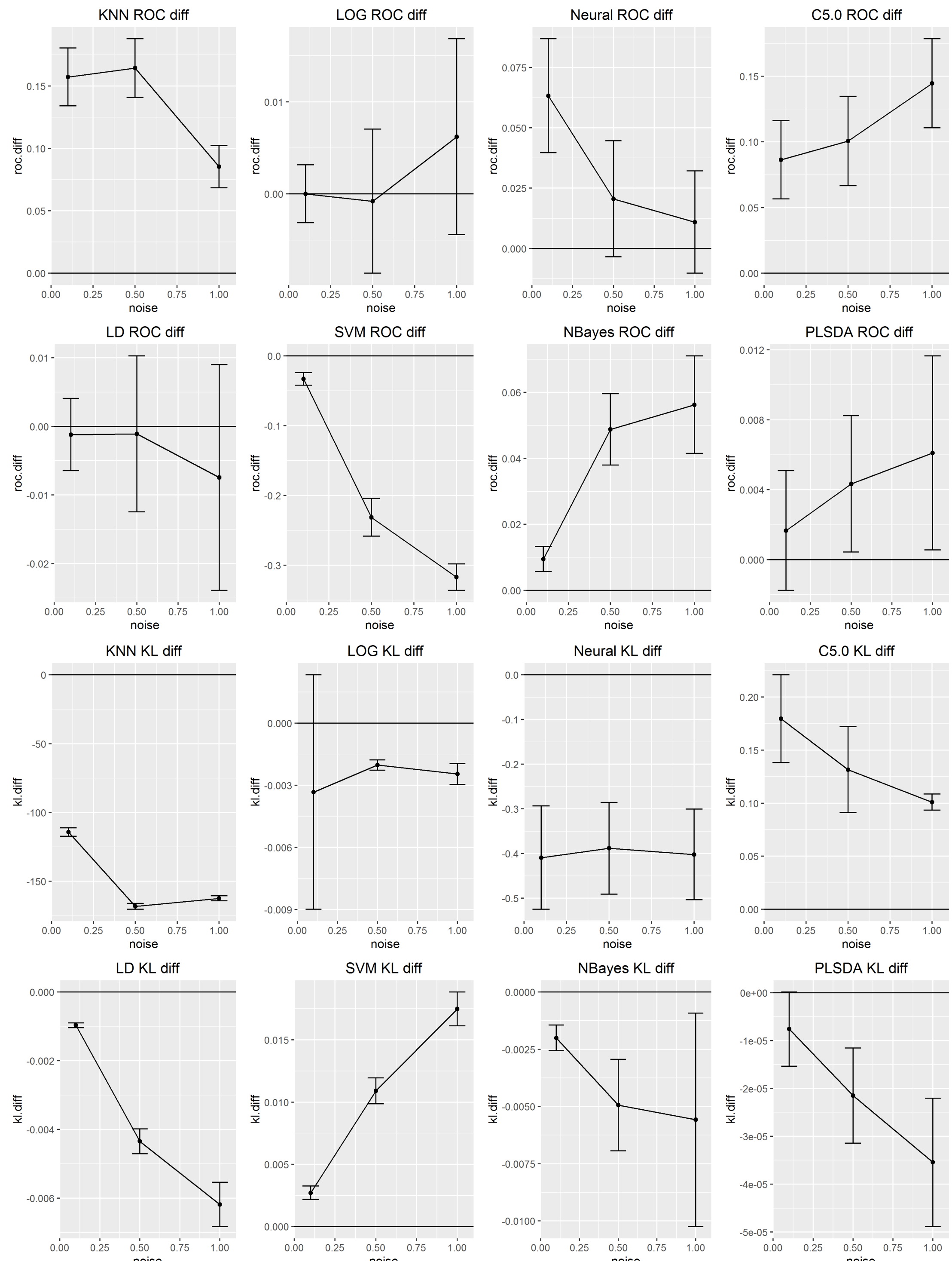


Figure 44: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 85.88

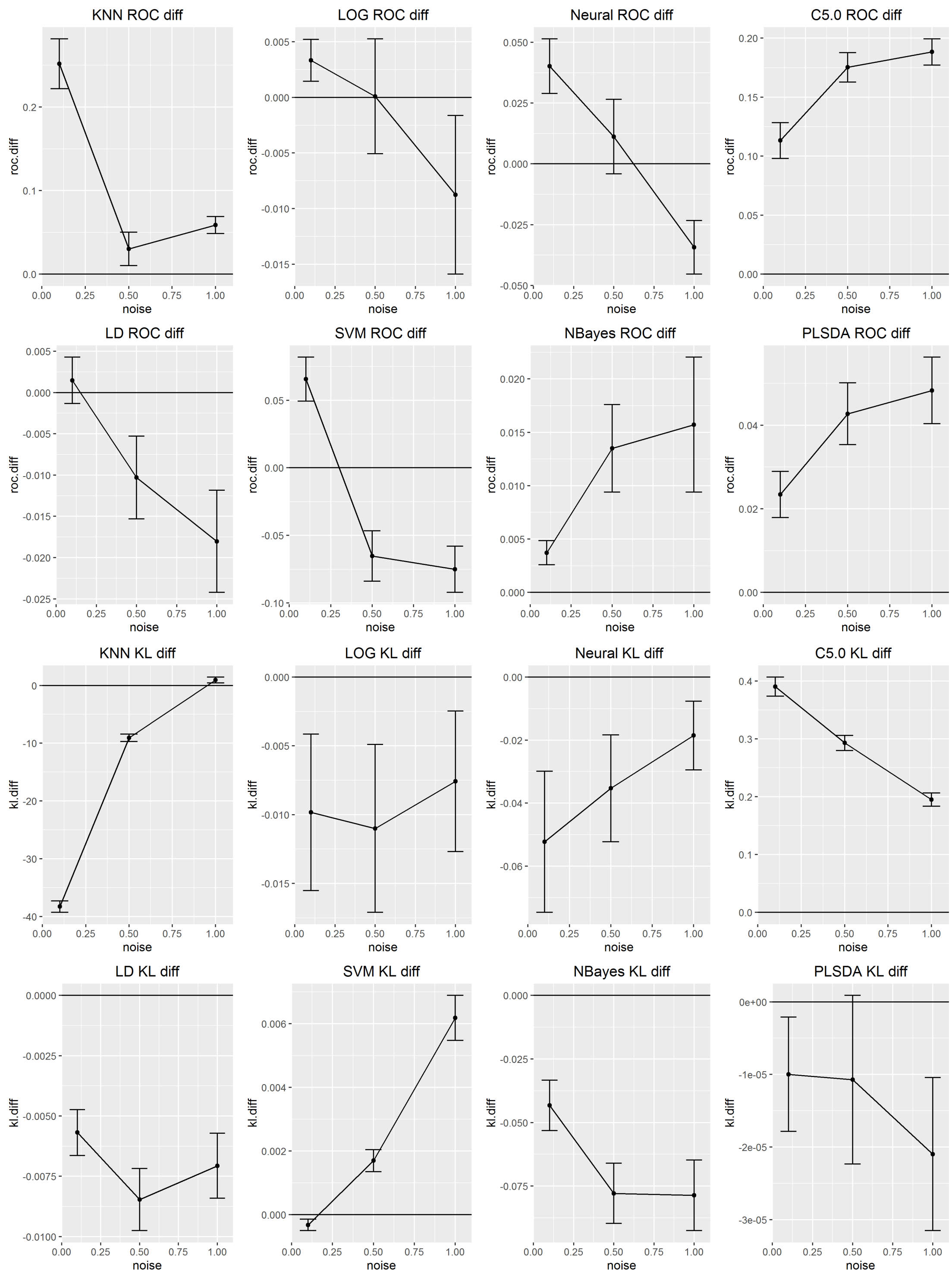


Figure 45: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the binary data set with IR = 129.44

C Outcomes for Multi-Class Data Sets

This section contains outcomes for all multi-class data sets in testing the noisy replication method. They are ordered by the imbalance ratio (IR). The first subgraph in each figure is the 95% confident intervals of △ROC, and the second is the 95% confident intervals of △KL distance after applying the noisy replication method.

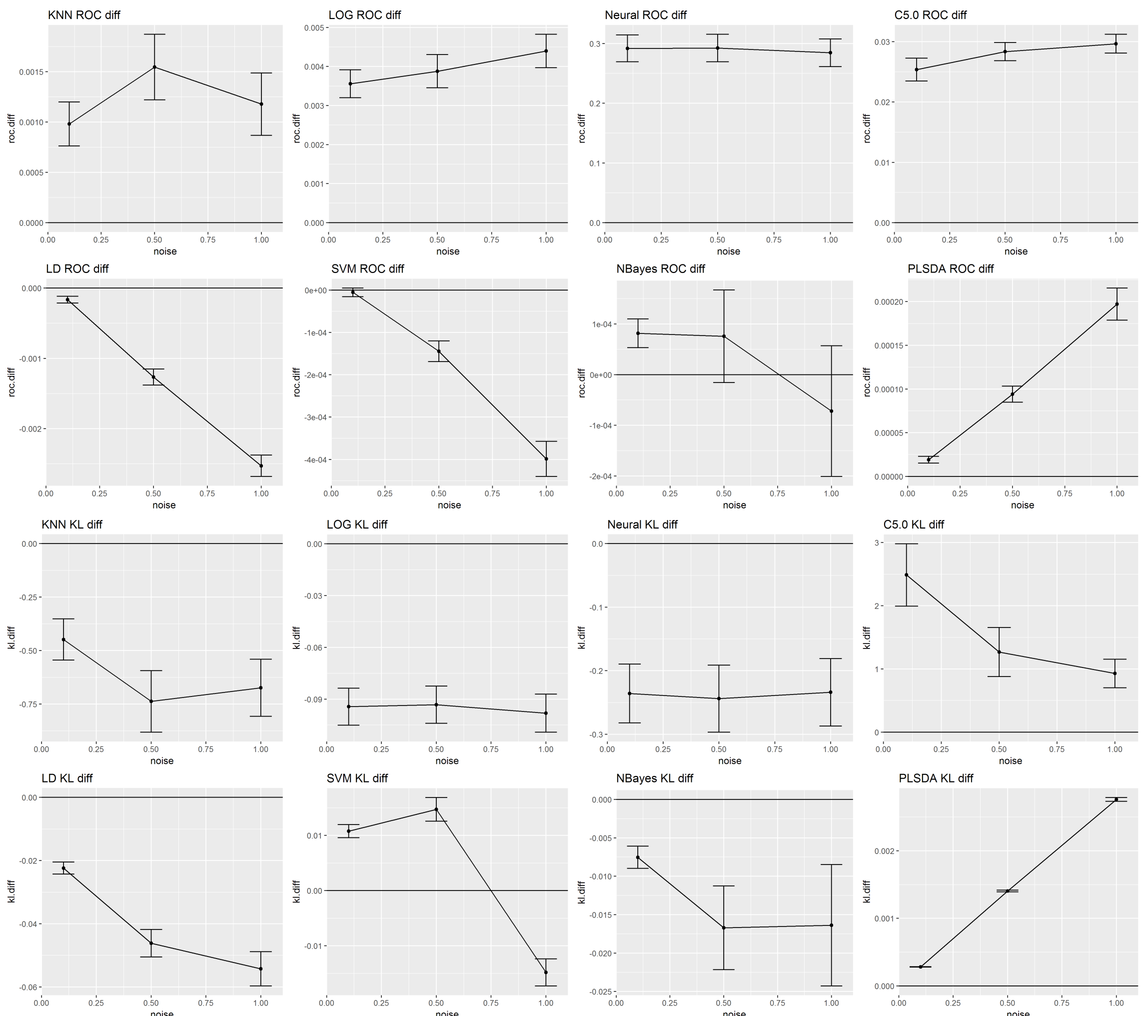


Figure 46: △ROC (top) and △KL-distance (bottom) outcomes for the multi-class data set with IR = 1.10

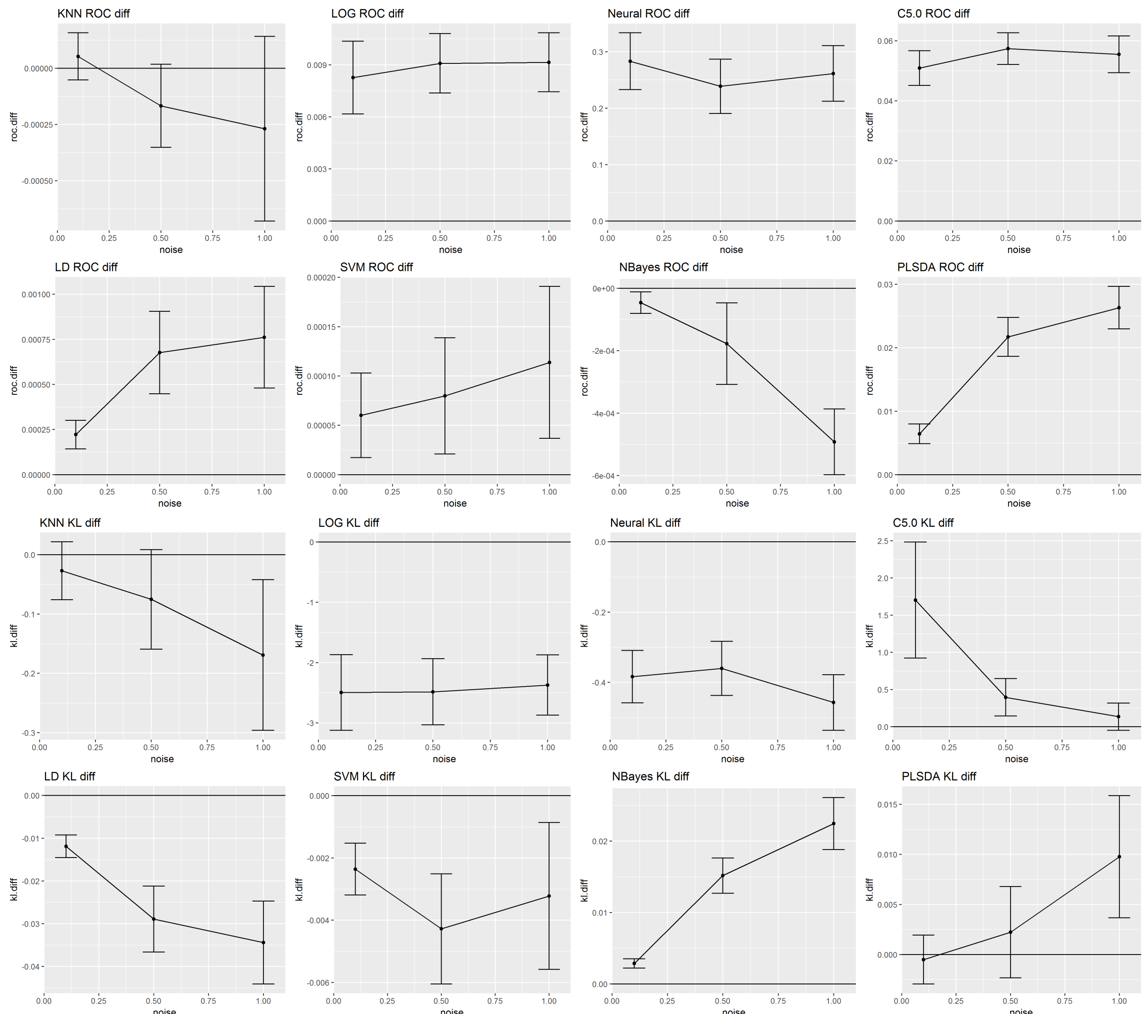


Figure 47: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the multi-class data set with IR = 1.48

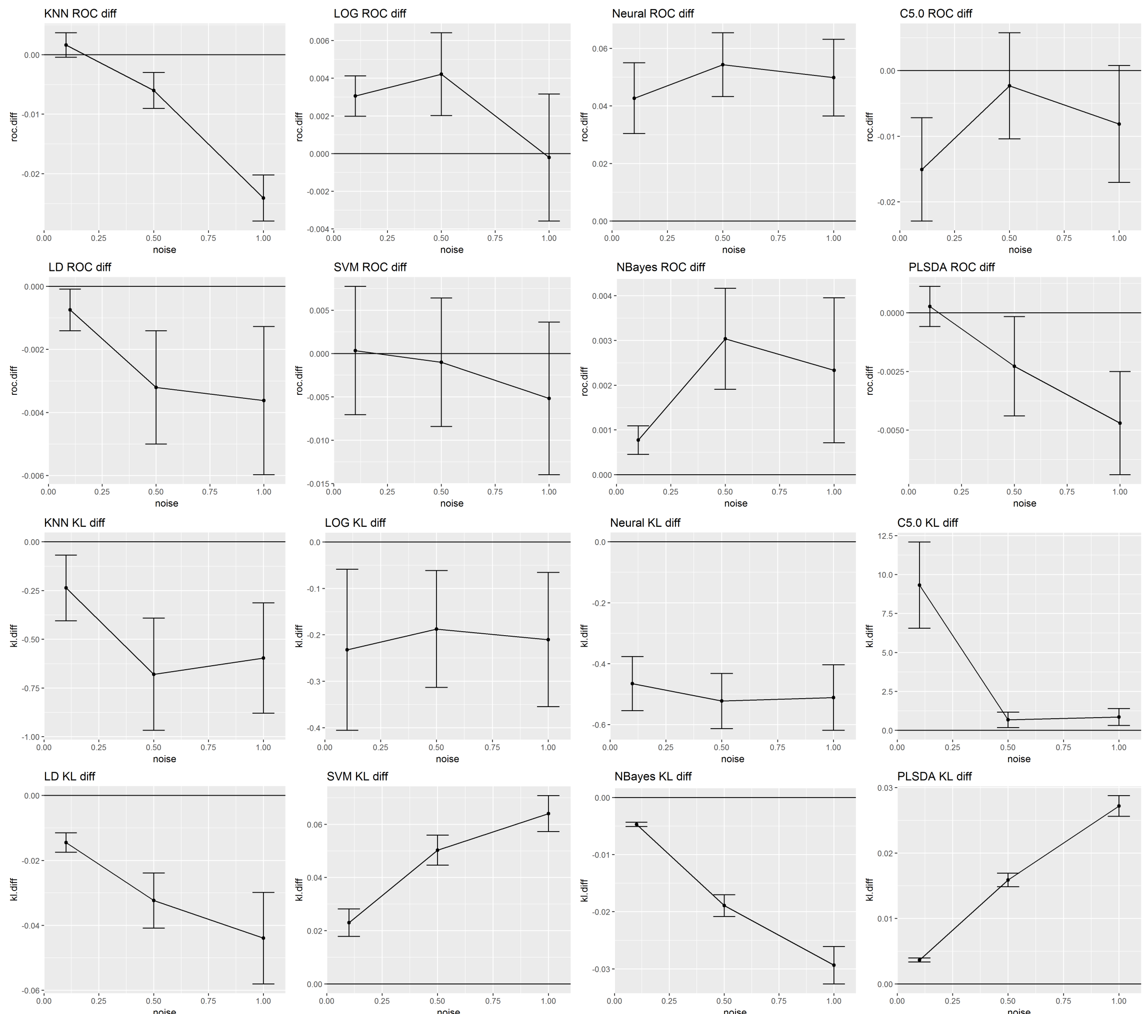


Figure 48: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the multi-class data set with IR = 1.70

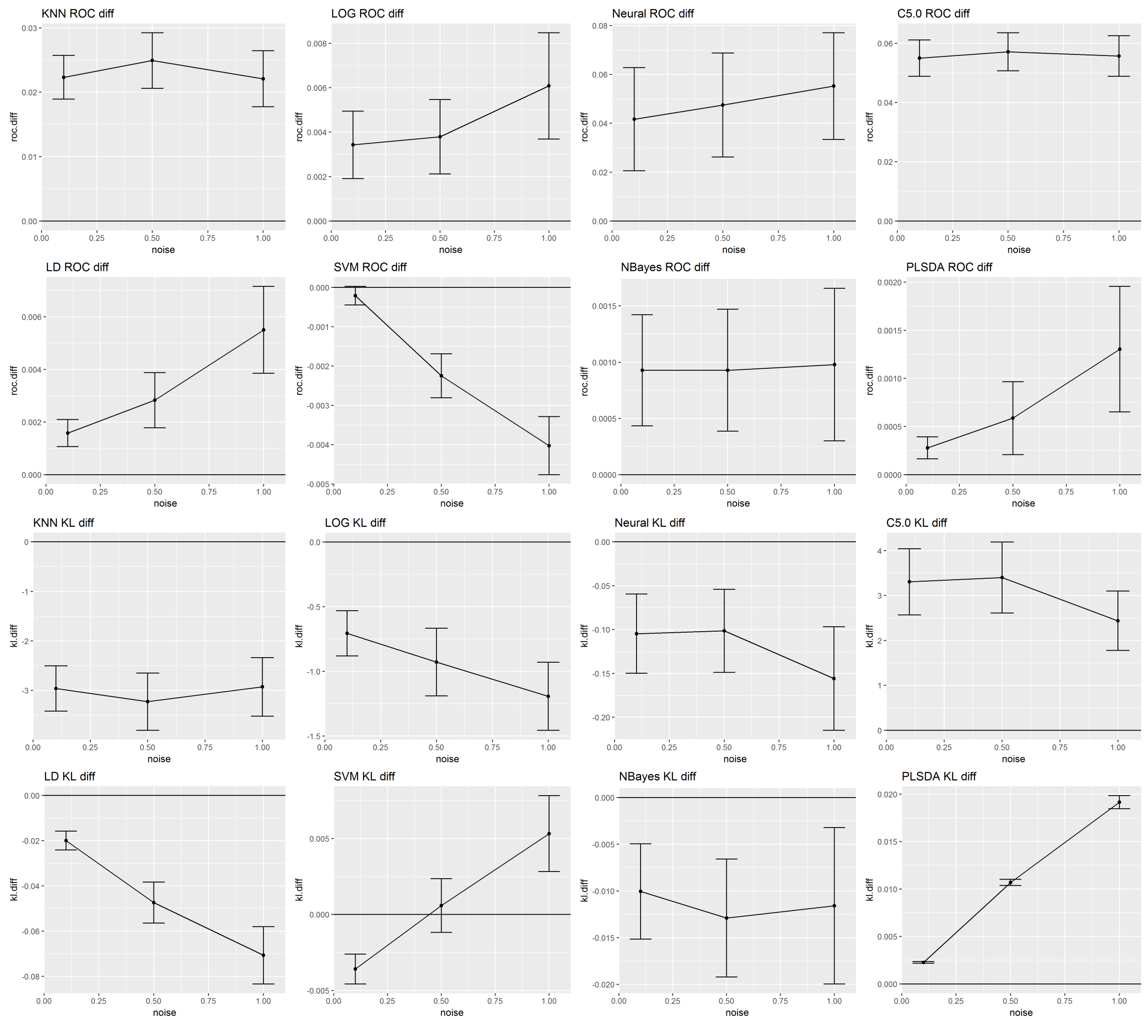


Figure 49: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the multi-class data set with IR = 5.00

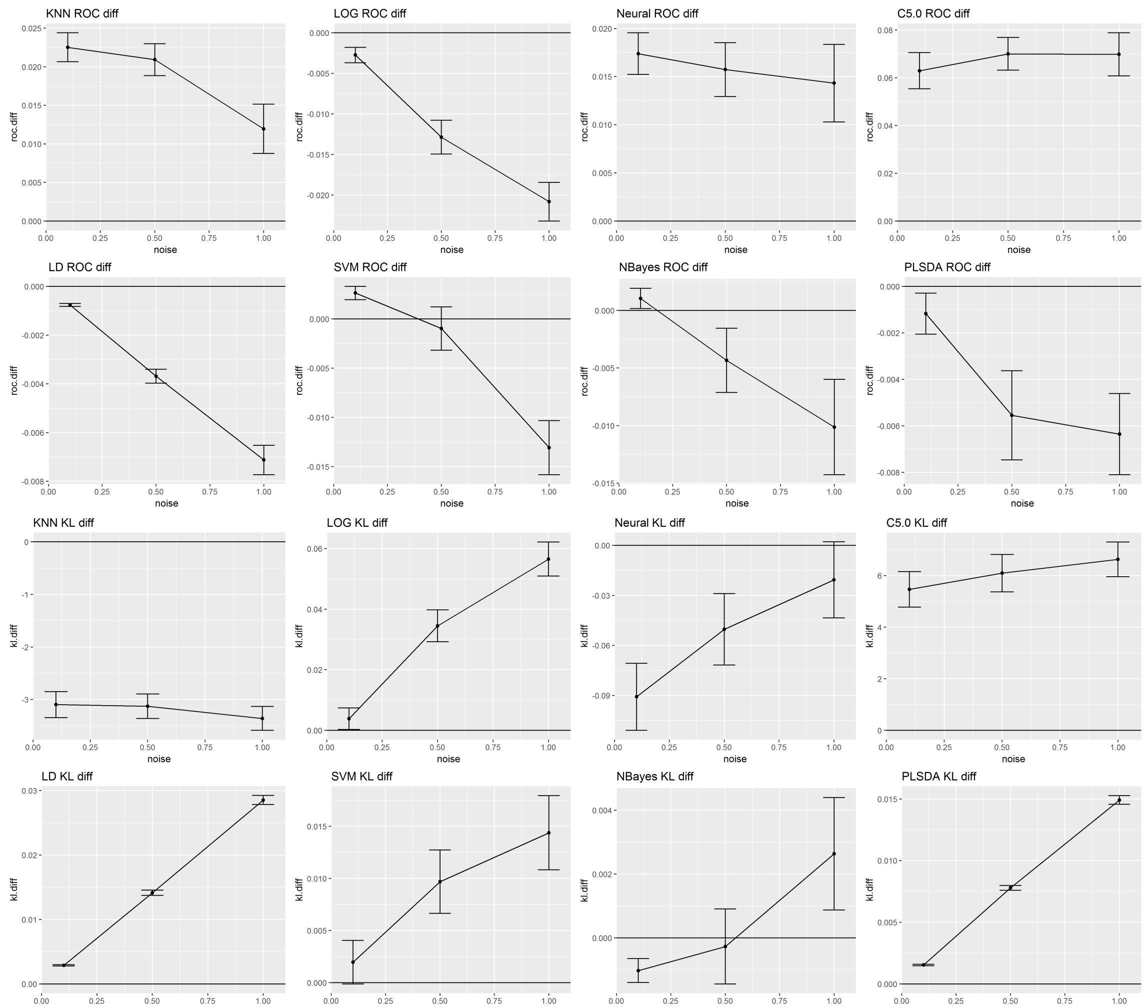


Figure 50: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the multi-class data set with IR = 5.88

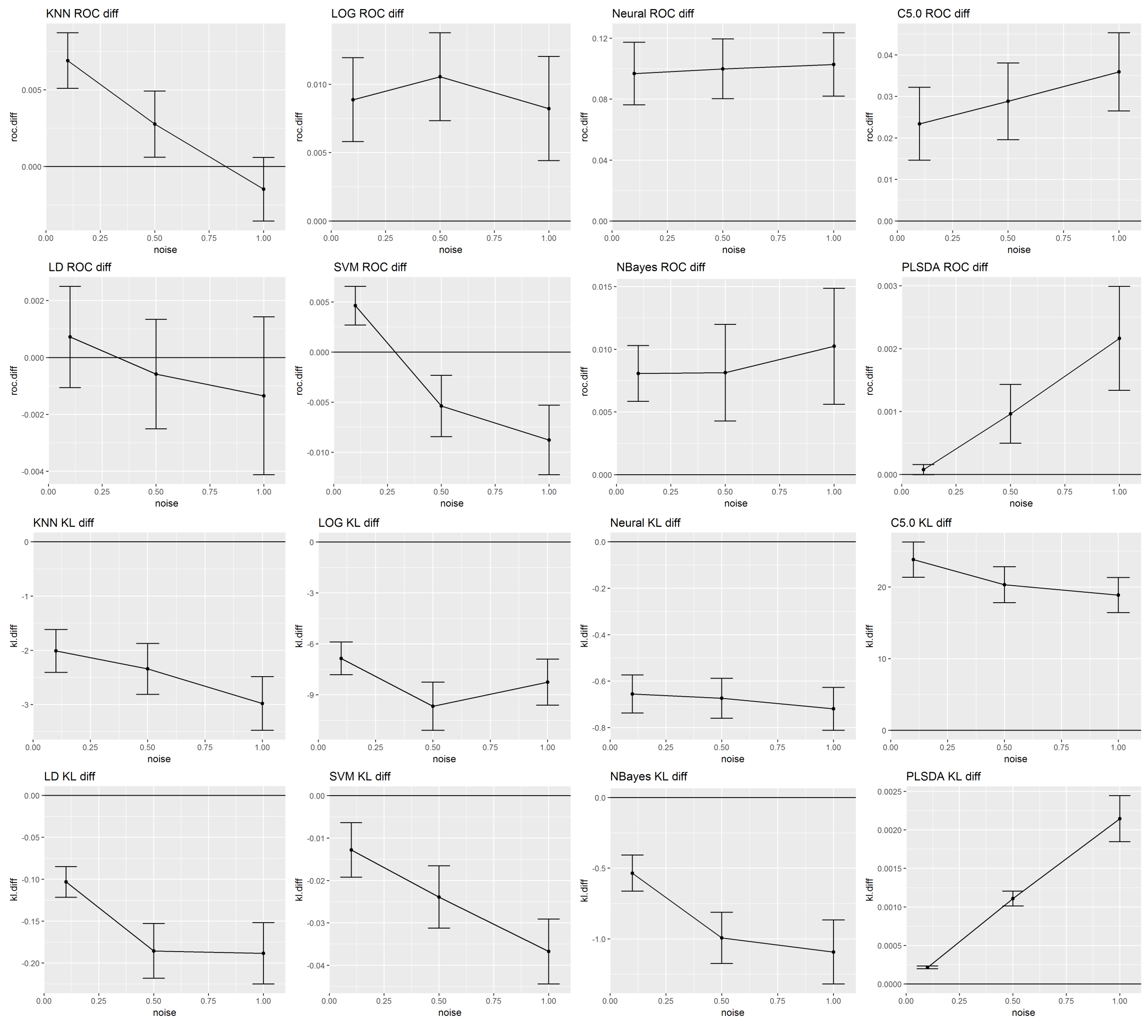


Figure 51: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the multi-class data set with IR = 8.44

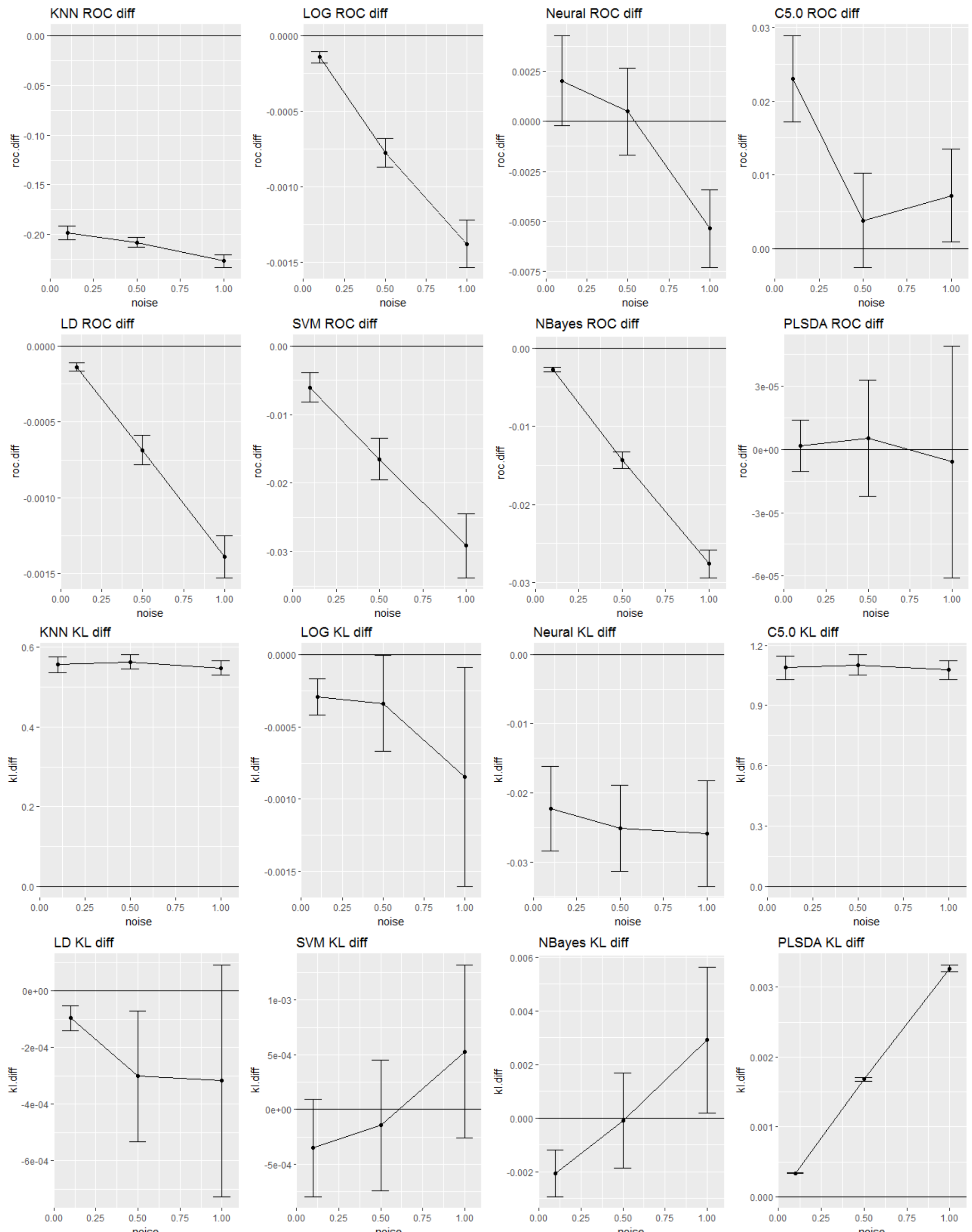


Figure 52: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the multi-class data set with IR = 9.00

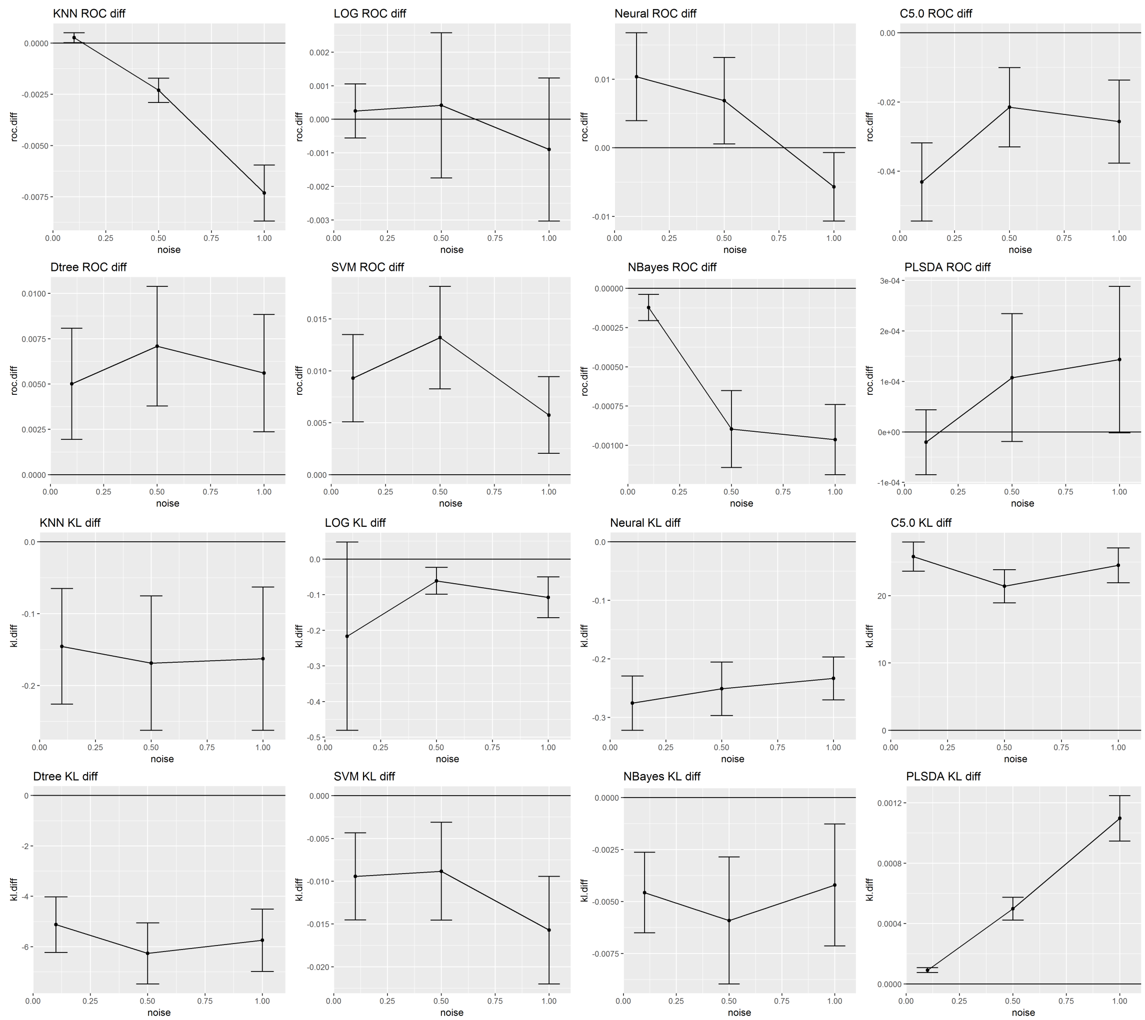


Figure 53: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the multi-class data set with IR = 28.60

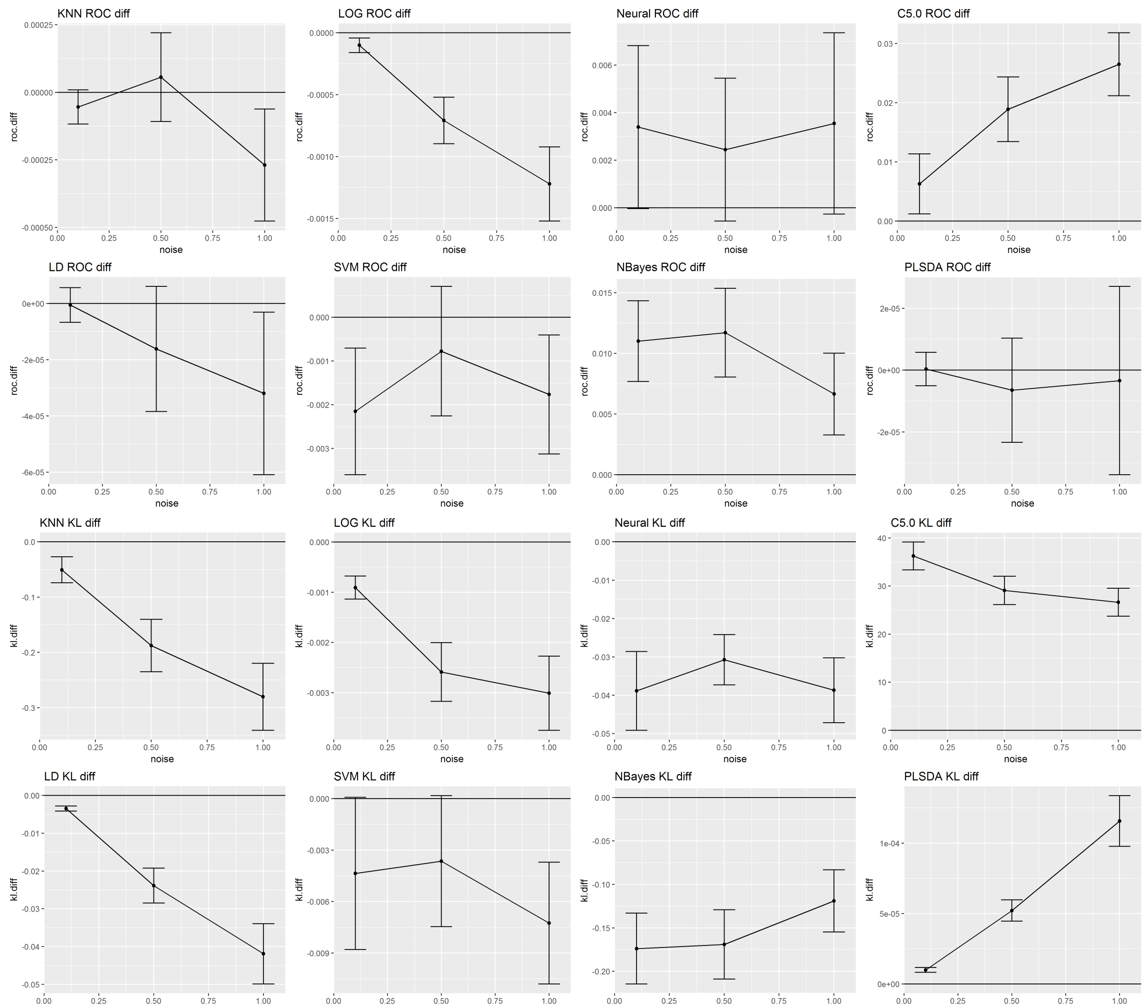


Figure 54: △ROC (top) and △KL-distance (bottom) outcomes for the multi-class data set with IR = 92.60

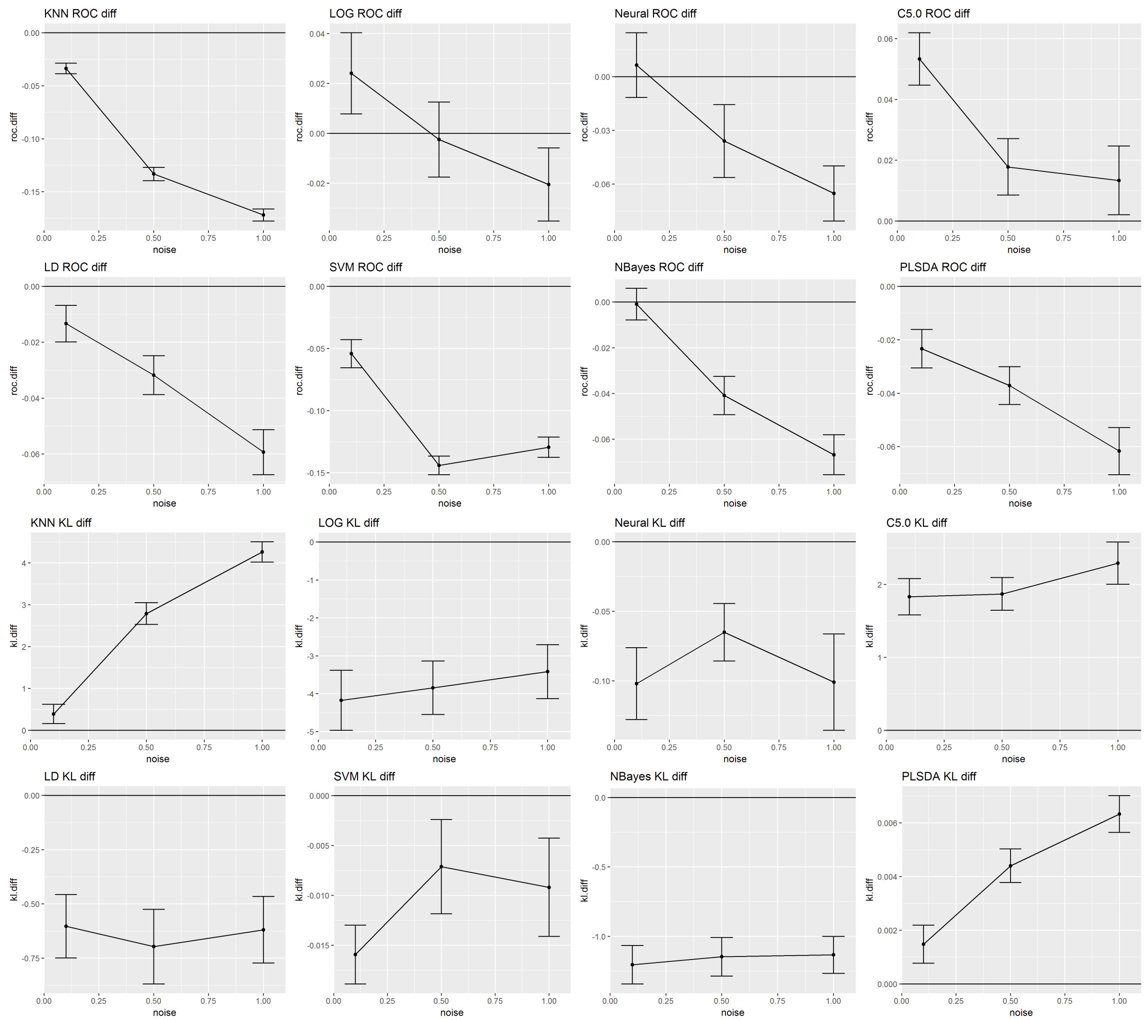


Figure 55: $\triangle$ROC (top) and $\triangle$KL-distance (bottom) outcomes for the multi-class data set with IR = 164.00

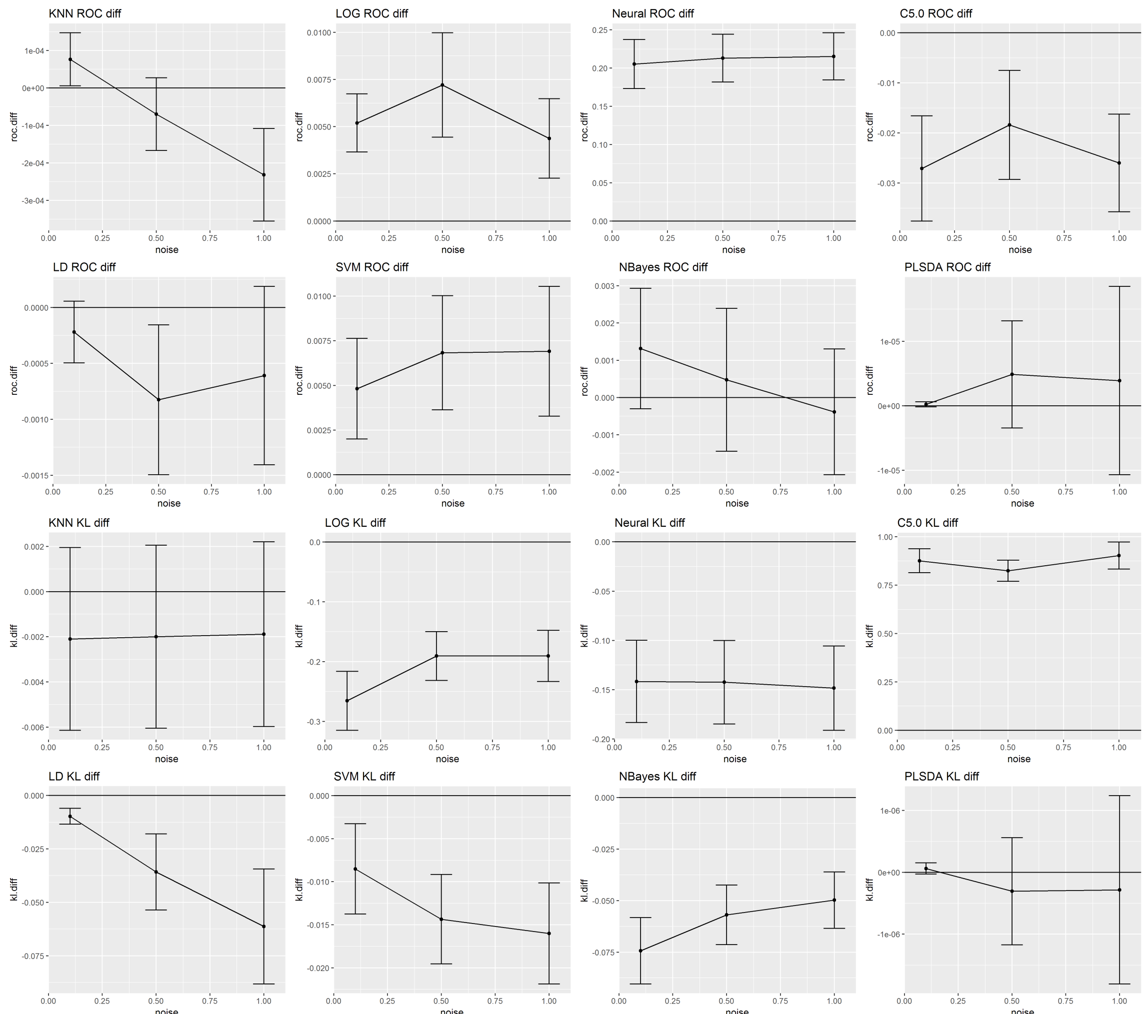


Figure 56: △ROC (top) and △KL-distance (bottom) outcomes for the multi-class data set with IR = 853.00