MSc Research Report

Comparative performance evaluation of logistic regression and machine learning methods on different data types

By

Zenzile B Ntshabele

Supervisors

Dr DM Rose and Dr C Chimedza

A research report submitted to the Faculty of Science, University of the Witwatersrand, in partial fulfilment of the requirements for the degree of Master of Science

October 7, 2022

Contents

Declaration

I declare that this research report was written by myself and that the work contained herein is my own except where explicitly stated otherwise in the text, that proper references have been provided to all supporting literature and resources, and that this work has not been submitted for any other degree or professional qualification.

07 October 2022

Acknowledgements

I would like to express my great appreciation to my supervisors, Dr David Rose and Dr Charles Chimedza, for their important contribution to the success of this project and my overall academic accomplishment. Thank you for your prompt response and patience throughout this process.

I would also like to thank my spouse who has been an incredible supporter and cheer-leader during this process, I am grateful to him.

List of Acronyms and Abbreviations

ANN	Artificial Neural Network
AUC	Area under the curve
kNN	k-nearest neighbors
LR	Logistic regression
RFE	Recursive feature elimination
RMSE	Root mean square error
ROC	Receiver operating characteristic
SVM	Support vector machine
WkNN	Weighted k-nearest neighbors

Glossary

Term	Definition
Accuracy	The ratio of correct predictions to total predictions
AUC	A numeric measure of how well the classifier can separate classes
Cohen's Kappa statistic	A measure of how well the classifier performed relative to how well it would have performed simply by chance
Kernel logistic regression	The kernel version of logistic regression that transfers the original input space into a high-dimensional feature space using kernel functions.
Negative likelihood ratio	The ratio of $(1-$ sensitivity) to the specificity
No Information Rate	The accuracy given all observations are assigned to the majority class
Precision	The ratio of correct positive predictions to the total predicted positives
ROC	Plots observed sensitivity versus $(1-$ specificity) for all possible classification thresholds. It measures the ability of the classifier to separate classes.
Sensitivity	The percentage of true positives correctly identified
Specificity	The percentage of the true negatives correctly identified
Youden's index	Measures the ability of a classifier to avoid failure and is expressed as; sensitivity$-(1-$ specificity)

Chapter 1

Introduction

1.1 Introduction

Different techniques have evolved in response to the need to solve prediction and classification problems. These techniques include traditional statistical methods and machine learning methods. Logistic regression (LR) is a commonly used statistical tool to predict a categorical response variable, in particular when the variable is binary (Musa, 2013). Its popularity can be attributed to its ability to produce excellent models that are simple and easy to interpret (Dreiseitl and Ohno-Machado, 2002). But lately, machine learning techniques have been applied in classification tasks. Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), and weighted k-nearest neighbors (WkNN) introduced by McCulloch and Pitts (1943), Cortes and Vapnik (1995) and Dudani (1976) respectively, are some of the machine learning methods used in classification and will be the focus of this study. The performance of these three machine learning methods will be compared with LR.

LR models are excellent at detecting possible causal relationships (Tu, 1996), but fall short at modelling when there are complex non-linear relationships between variables (Eftekhar et al., 2005). In contrast, ANNs tend to perform well when the relationships between variables are complex, multidimensional, and non-linear (Tu, 1996). However, due to the black-box nature of ANNs, the model parameters tend to be too

complex to interpret (Intrator and Intrator, 2001).

SVMs perform well even in high dimensional spaces (Doran et al., 2007), avoid overfitting better than ANNs (Min and Lee, 2005) and are flexible in their ability to avoid outliers (Doran et al., 2007). However, SVMs take longer to train than ANNs for large datasets with many support vectors (Byun and Lee, 2003). The kNN classifier is one of the simple machine learning methods widely applied in pattern recognition and classification problems (Rizwan and Anderson, 2016). WkNN is an extension of the kNN where the observations within the training set, which are particularly close to the new observation, are assigned higher weights than those further from the new observation (Zhao and Chen, 2016).

Based on the literature, comparisons of these techniques have been performed numerous times. The objectives of this study are to compare the performance of LR with ANNs, SVMs, and WkNN under a variety of conditions using simulations.

1.2 Motivation

LR, ANNs, SVMs, and WkNN have been effective in solving prediction and classification problems. However, it is not always easy to decide which technique to use or which will produce better results as their performance is data-dependent (Kim, 2008).

The performance of these techniques has been compared using specific data, and their performance differs with different datasets. Li et al. (2001) applied ANN and regression to a complex wind power generation dataset, and due to the non-linearity in the data, ANN outperformed the regression model. Huang et al. (2005) used the SVM, linear discriminant analysis, quadratic discriminant analysis, and ANNs to predict the financial market price movement; the SVM outperformed all the competing models.

Kim (2010) used simulated data to compare the performance of the ANN, LR, and a decision tree based on the number of classes of the response variables, the number of predictors, the types of predictors, the number of classes of predictors, and the sample

size. He found that LR performed better than the competing methods in the case of a binary response variable with continuous predictors. Its performance was also better for continuous and categorical predictors given a small number of predictor variables and a small sample size. ANNs performed better as the number of predictors and the number of classes of the response variable increased.

This study hopes to build on Kim (2010)'s work by evaluating the performance of the LR, ANNs, SVMs, and WkNN over different simulated scenarios detailed in Section 1.3.

1.3 Aim and Objectives

1.3.1 Aim

The main aim of this investigation is to observe how the LR model performs in comparison with machine learning models such as ANNs, SVMs, and WkNN. Models are applied to various data types generated using simulation methods.

1.3.2 Objectives

The objectives of this study are to;

1. Investigate the impact of binning predictors on the model performance
2. Determine the impact of skewness in predictors on the model performance
3. Assess the impact of data dimensions on the model performance
4. Assess how a model performs when less important features are removed from the model
5. Assess the effect of using different logistic regression link functions
6. Assess the effect of imbalance in the response categories.

1.3.3 Significance of the Study

As described in Section 1.2, knowing which technique will produce better results under specific conditions is not always easy. It is also difficult to compare their performance and determine the best one since that depends on the data. This study hopes to add to the literature by investigating the performance of machine learning and statistical methods using simulated datasets. This should then give guidance on the method to employ for different datasets depending on whether the interest is in the prediction or interpretation of causal relationships.

Chapter 2

Literature and Theoretical Review

2.1 Literature Review

This section reviews the literature on a comparison of the performance of LR, ANNs, SVMs, and WkNN.

SubbaNarasimha et al. (2000) compared the performance of ANNs and multiple regression when the data contained skewed response variables. The results of the study showed lower error scores for the multiple regression model than those of the ANN. This study concluded that multiple regression models perform better than ANN models even when data have skewed response variables once appropriate transformations are applied to the skewed variables. Unlike SubbaNarasimha et al. (2000), this study will focus on skewed predictors and unbalanced response classes.

Khan et al. (2001) applied ANNs to the classification and diagnosis of cancers and found that ANNs performed well. The case of a skewed response was studied by Kumar (2005) where the performance of regression and ANN was compared. The regression model performed better than ANNs for skewed data and the Root Mean Square Error (RMSE) was used as the performance measure. This study examines the effect of unbalanced response classes in a classification problem, as opposed to skewed response, which is a regression problem. In Min and Lee (2005), SVMs performed better

than the LR, ANN, and a discriminant analysis model when applied to the problem of bankruptcy prediction.

Boyacioglu et al. (2009) applied ANNs, SVMs, and multivariate statistical methods (Multivariate discriminant analysis, K-means cluster analysis and LR) to a bank financial failure prediction problem. The classification performance of the techniques was evaluated using accuracy. A multi-layer perceptron neural network was superior and SVM performed better than most of the other competing techniques.

Muniz et al. (2010) compared LR, ANN, and an SVM for classifying Parkinson's disease and assessing the effects of the treatment. The performance of the models was evaluated using the AUC. The ANN outperformed LR and SVM according to the AUC and the negative likelihood ratio. However, an SVM gave the best accuracy.
LR, ANNs, SVMs, and multivariate discriminant analysis were used to predict hotel bankruptcy (Kim, 2011). The performance of the models was evaluated using the overall classification error, accuracy, and relative error cost ratios. The study showed that both ANNs and SVMs provided more accurate estimates, resulting in higher classification rates than other competing models. Although there was no significant difference in prediction accuracy between ANNs and SVMs, the ANN model was a better predictor.

Lin et al. (2011) investigated the ability of LR, auto-logistic regression (a LR model that takes spatial autocorrelation into account), and ANN models to detect land-use changes and the relationships between land use and its drivers. AUC, Kappa statistic, multiple resolution validation, and landscape metrics were used as performance measures for the three techniques. ANNs gave better results relative to the LR and auto-logistic models.

The study by Musa (2013) applied LR and SVMs to various sizes of balanced and unbalanced datasets using bagging and ensemble methods. Several measures were used to evaluate performance; these included accuracy, sensitivity, specificity, precision, the area under the ROC curve (AUC), and so on. The study showed that the SVM and LR overall performed equally well for balanced and unbalanced data. An SVM was shown

to perform better for highly unbalanced datasets.

LR, ANNs, naive Bayes and random forests were used to predict mortality from injuries caused by burns (Stylianou et al., 2015). Among other metrics, the AUC and Youden's index were used to evaluate model performance. The results showed no significant differences between the performance of the LR and machine learning methods. ANNs overall gave a better prediction, and LR gave a good balance of performance and interpretability. It was concluded that the LR model gives a comparable performance to the more complex machine learning methods in predicting in-hospital mortality among burn patients.

Bui et al. (2016) compared a kernel LR (a LR model that transforms the input space into a high-dimensional feature space using kernel functions), ANNs, SVM, and a decision tree for landslide susceptibility modelling. Models were compared using the AUC, Kappa statistic, accuracy, sensitivity, specificity, and so on. Overall, the multilayer perceptron neural network gave better prediction, followed by the SVM. Results revealed that both the kernel LR model and a decision tree were promising methods for shallow landslide susceptibility mapping.

El_Jerjawi and Abu-Naser (2018) used ANNs to predict diabetes and achieved an accuracy level of 87% with a reasonably small average error while Mohsen et al. (2018) also used ANNs for the classification of a brain tumor, and the ANN performed better than the other classifiers including kNN and linear discriminant analysis.

According to Salas-Eljatib et al. (2018), an imbalance in response categories increases the variability and bias of the estimated parameters when fitting a LR model. The results showed that the highest precision of all estimated parameters and the lowest RMSE occurred with balanced data. Salas-Eljatib et al. (2018) further revealed that the logit model prediction capability was greatly affected by the imbalance in data, evident in the overall error (increasing with an increase in the level of imbalance) and accuracy (decreasing with the increase in the level of imbalance).

Inayah et al. (2020) compared the performance of kNN and WkNN in predicting the interaction of HIV proteins with human proteins. Accuracy, sensitivity, specificity, and precision were used to measure performance, and the WkNN performed better than the kNN.

WkNN was used to classify patients with epilepsy (Wang et al., 2020). Also, the performance of the WkNN was compared to that of an ANN, SVM, kNN, decision tree, random forest, and the extreme gradient boosting. Using the accuracy, sensitivity, specificity, and false alarm rate as performance measures the WkNN outperformed all the other classification methods.

2.2 Theoretical Review

This section reviews the theory behind the techniques that will be used.

Let $L = (\mathbf{x}_i, y_i)$, $i = 1, 2.., n$ represent a training dataset where $\mathbf{x}_i$ denotes a $1 \times m$ vector of inputs and $\mathbf{y}$ is a $n \times 1$ vector of outputs where $y_i \in \{0, 1\}$ denotes a binary response variable. In addition, assume that $Prob(y_i = 1|\mathbf{x}_i) = p(\mathbf{X})$ is a probability of success, and $\mathbf{X}$ is a $n \times m$ matrix of all inputs.

2.2.1 Logistic Regression

LR is a statistical method that estimates the probability of a binary response variable in relation to a set of predictors that can either be numeric or categorical (Izenman, 2008). For a given training dataset, the probability of success is given by;

$$p(\mathbf{X}) = \frac{1}{1 + e^{-\boldsymbol{\beta}^{\tau}\mathbf{X}}} \tag{2.1}$$

Where $\boldsymbol{\beta}$ is a $m \times 1$ vector of coefficients, after manipulation, Equation 2.1 yields the odds given by:

$$\frac{p(\mathbf{X})}{1 - p(\mathbf{X})} = e^{\beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \cdots + \beta_m x_{m,i}} \tag{2.2}$$

Odds can take any positive value and the values of the odds close to 0 and infinity indicate very low and very high probabilities of occurrence (i.e. $y_i = 1$) respectively. Taking the logarithm of both sides of Equation 2.2 yields;

$$\log\left(\frac{p(\mathbf{X})}{1-p(\mathbf{X})}\right) = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \cdots + \beta_p x_{m,i} \tag{2.3}$$

The left-hand side of Equation 2.3 is referred to as the log-odds or logit and the model parameters $\beta_0, \beta_1, ..., \beta_m$ are estimated using the Maximum Likelihood Method.
Using the Bernoulli density function, the likelihood of an observation (y_i, x_i) is given by:

$$L(\boldsymbol{\beta}; y_i; x_i) = p(x_i; \beta_i)^{y_i}(1 - p(x_i; \beta_i))^{1-y_i}$$

Assuming that observations are independent and identically distributed, the maximum likelihood function of the dataset is given by:

$$L(\boldsymbol{\beta}; \mathbf{y}; \mathbf{X}) = \prod_{i=1}^{n} p(\mathbf{x}_i)^{y_i}(1 - p(\mathbf{x}_i))^{1-y_i}$$

Taking the logarithm, then the log-likelihood becomes

$$l(\boldsymbol{\beta}; \mathbf{y}; \mathbf{X}) = \sum_{i=1}^{n} [y_i \ln p(\mathbf{x}_i) + (1 - y_i) \ln(1 - p(\mathbf{x}_i))]$$

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} [y_i \boldsymbol{\beta}^\tau \mathbf{x}_i - \ln(1 + e^{\boldsymbol{\beta}^\tau \mathbf{x}_i})]$$

Assuming that the vector of predictors $\mathbf{x}_i$ includes the constant term 1 to accommodate the intercept, to maximise the log-likelihood, we set its derivatives to zero. Then the vector of first derivatives of the log-likelihood with respect to the parameter $\boldsymbol{\beta}$, is

$$\frac{\partial l(\boldsymbol{\beta}; \mathbf{y}; \mathbf{X})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} \mathbf{x}_i (y_i - p(\mathbf{x}_i; \boldsymbol{\beta})) = 0 \tag{2.4}$$

Equations 2.4 is solved using the Newton–Raphson algorithm, which requires a Hessian matrix, which is a matrix of second derivatives given by

$$\frac{\partial^2 l(\boldsymbol{\beta}; \mathbf{y}; \mathbf{X})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\tau} = -\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\tau p(\mathbf{x}_i; \boldsymbol{\beta})(1 - p(\mathbf{x}_i; \boldsymbol{\beta}))$$

The method starts from a guess of the solution $\hat{\boldsymbol{\beta}}_0$, and then recursively update the guess with the equation

$$\hat{\boldsymbol{\beta}}_t = \hat{\boldsymbol{\beta}}_{t-1} - \left[\frac{\partial^2 l(\hat{\boldsymbol{\beta}}_{t-1}; \mathbf{y}; \mathbf{X})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\tau}\right]^{-1} \left[\frac{\partial l(\hat{\boldsymbol{\beta}}_{t-1}; \mathbf{y}; \mathbf{X})}{\partial \boldsymbol{\beta}}\right]$$

Until numerical convergence of $\hat{\boldsymbol{\beta}}_t$ to the solution $\hat{\boldsymbol{\beta}}$

2.2.2 Artificial Neural Networks

ANNs assume no prior distribution of the data but learn from the input variables (Li et al., 2001). There are different ANN architectures to perform various tasks. A multi-layer perceptron is one of the widely applied ANN in classification or regression tasks (Mohsen et al., 2018). It is a typical multi-layer feedforward ANN that non-linearly maps an input vector $\mathbf{X} = (X_1, X_2, \ldots, X_m)^\tau$ of predictors to an output vector $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_u)^\tau$ of variables, where m is the number of input nodes, u is the number of output nodes, and τ denotes transposition. Between the inputs and the outputs, there are also hidden variables called neurons or nodes arranged in layers. If the ANN has l hidden layers then it is referred to as a $(l+1)$-layer ANN. The rth output node is expressed as;

$$Y_r = \boldsymbol{\Phi}_r(\mathbf{X}) + \epsilon_r, \qquad r = 1, 2 \ldots, u$$

where

$$\boldsymbol{\Phi}(\mathbf{X}) = \mathbf{g}(\boldsymbol{\theta_0} + \mathbf{D}\mathbf{f}(\boldsymbol{\beta_0} + \mathbf{H}\mathbf{X})). \tag{2.5}$$

Here $\mathbf{H} = (\beta_{ij})$ is a $s \times m$-matrix of weights between the input nodes and the hidden layer of s hidden nodes, $\mathbf{D} = (\theta_{jr})$ is a $u \times s$-matrix of weights between the hidden layer and the output layer, $\boldsymbol{\beta}_0 = (\beta_{01}, \beta_{02}, ..., \beta_{0s})^\tau$, $\boldsymbol{\theta}_0 = (\theta_{01}, \theta_{02}, ..., \theta_{0u})^\tau$ are s and u vectors of biases respectively; $\mathbf{f} = (f_1, f_2, \ldots, f_s)^\tau$ and $\mathbf{g} = (g_1, g_2, \ldots, g_u)^\tau$ are the vectors of non-linear activation functions.

In estimating a vector consisting of the parameters of a fully connected network, the error sum of squares is minimised.

$$\text{SSE}(\boldsymbol{\omega}) = \sum_{i=1}^{n} \sum_{r \in \kappa} (Y_{i,r} - \hat{Y}_{i,r})^2$$

where $\boldsymbol{\omega}$ is a vector consisting of the connection weights (matrices $\mathbf{D}$ and $\mathbf{H}$ in Equation 2.4 and the biases (the vectors $\boldsymbol{\theta}_0$ and $\boldsymbol{\beta}_0$). κ is a set of output nodes, $Y_{i,r}$ is the value of the true output u-vector, and $\hat{Y}_{i,r}$ is the value of the fitted output u-vector. In classification, the number of output nodes corresponds to the number of categorical classes. The ANN parameters are then estimated using the backpropagation algorithm, which is an algorithm used to train a feedforward network (Izenman, 2008).

2.2.3 Support Vector Machines

A SVM is a machine learning method based on statistical learning theory that transforms original input space into a higher-dimensional feature space to find an optimal separating hyperplane (Izenman, 2008). This technique has been successfully applied to a diverse range of classifications such as handwritten digit recognition (Sadri et al., 2003), text categorization (Joachims, 1998), and financial forecasting (Tay and Cao, 2001).

SVM use kernel functions to transform decision boundaries. Kernel functions provide an efficient method for transforming data into higher dimensions, and their use is not restricted to the SVM; these will also be implemented in the WkNN in the following section. Given a training dataset L, and the kernel function K, SVM finds the optimal α_i for each x_i to maximise the margin between the hyperplane and the closest observations to it. The class prediction for a new observation $\mathbf{x}$ from the test set is made through:

$$\text{sign}\left(f(\mathbf{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i)\right), \qquad \forall i$$

where β_0 is a threshold and $y_i \in \{0, 1\}$. There are different forms of kernel functions; this includes the Sigmoid kernel, Gaussian radial function, and Polynomial kernels. The non-linear optimization problem finds the value of a vector $\boldsymbol{\alpha}$ that maximises the following function:

$$L = \frac{||\boldsymbol{\beta}||^2}{2} + C\sum_{i=1}^{n}\epsilon_i - \sum_{i=1}^{n}\alpha_i[y_i(\beta_0 + \mathbf{x}_i^T\boldsymbol{\beta}) - 1 + \epsilon_i] - \sum_{i=1}^{n}\epsilon_i g_i,$$

subject to

$$0 \leq \alpha_i \leq C \quad \text{and} \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \qquad \text{for} \quad \alpha_i \geq 0 \quad \text{and} \quad g_i \geq 0$$

where α_i is the weight assigned to x_i, $\boldsymbol{\beta}$ is a normal vector for the separating hyperplane, also known as the margin. C is a penalty constant used to trade-off between the empirical error $\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2, \ldots, \epsilon_n)$ and the margin. The empirical error indicates the proportional amount by which the prediction is on the wrong side of its margin, and if $\alpha_i > 0$, then x_i is called a support vector.

2.2.4 Weighted k-Nearest Neighbors

This classifier places a new observation into the class that appears most frequently among the k-nearest neighbors based on the distance measures (Zhao and Chen, 2016). It involves determining the value of the k (nearest neighbors) parameter. Small values of k may lead to overfitting and large values of k may cause the neighborhood to include too many observations from other classes (Zhao and Chen, 2016). k-fold cross-validation is one of the commonly used methods to select an optimal value of k (Khuman, 2013).

Thereafter, the distance between the observations in the test dataset and those in the training dataset is calculated using distance functions. Such functions include the Euclidean distance, Mahalanobis distance, and Chebyshev distance. Given the training dataset L, for a new observation (x, y) the nearest neighbor (x_1, y_1) within the training dataset is determined by Equation 2.6 below.

$$d(x, x_1) = \min_i(d(x, x_i)) \tag{2.6}$$

where Equation 2.6 is a distance function and $\hat{y} = y_i$, the class of the nearest neighbor is selected as a prediction for y. Lastly, weights are allocated to the test dataset according to any arbitrary kernel function which transforms distance into weight in this

case. These are functions $K(.)$ of the distances d and must satisfy the conditions below (Hechenbichler and Schliep, 2004).

1. $K(d) \geq 0$ for all $d \in \Re$
2. $K(d)$ reaches its maximum at $d = 0$
3. $K(d)$ decreases monotonically as $d \to \infty$

The kernel functions include the following:

1. Rectangular kernel: $\frac{1}{2} I(|d| \leq 1)$ where I is the indicator function
2. Triangular kernel: $(1 - |d|) \cdot I(|d| \leq 1)$
3. Inversion kernel: $\frac{1}{|d|}$

2.3 Assessment Tools

Different performance metrics measure different tradeoffs in the classification performed by the models. Since the model can perform well on one metric and poorly on another, this study evaluated model performance on a variety of metrics.

The accuracy (a ratio of correct predictions to total predictions), sensitivity (the proportion of true positives correctly identified), and specificity (the proportion of true negatives correctly identified), computed from the confusion matrix (Pearson, 1904), were used as performance measures. These metrics' values range between 0 and 1 (or 0 to 100 if expressed as a percentage), and the closer they are to 1, the better. Given that TP and TN indicate the number of true positives and true negatives, respectively, and that FP and FN indicate the number of false positives and false negatives, respectively, the accuracy, sensitivity, and specificity are computed as follows:

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

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

In addition, the study used the AUC as one of the metrics. The ROC is a probability curve that plots sensitivity against (1-specificity) for all possible classification thresholds, and the AUC is an aggregate numeric measure of performance across all these thresholds. The AUC assesses how well the classifier can separate classes and is given by Bradley (1997) as:

$$AUC = \sum_i \left[Sens_i.\Delta(1 - Spec_i) + \frac{1}{2}\Delta Sens.\Delta(1 - Spec) \right]$$

Where

$$\Delta Sens = Sensitivity_i - Sensitivity_{i-1}$$

$$\Delta(1 - Spec) = [(1 - Specificity_i) - (1 - Specificity_{i-1})]$$

The Kappa statistic (Cohen, 1960) was a critical performance measure since it accounts for the possibility of agreement occurring by chance, which was particularly relevant given that this study investigated the effect of unbalanced data on model performance. This statistic measures how well the classifier performed relative to how well it would have performed simply by chance. If there is a perfect agreement between a classifier and a fact when classifying observations, then the Kappa statistic has a value of 1, and a value of 0 if there is no agreement beyond what would be expected by chance. The Kappa statistic can be calculated as:

$$Kappa\ statistic = \frac{2(TP \times TN - FN \times FP)}{(TP + FP)(FP + TN) + (TP + FN)(FN + TN)}$$

Cohen (1960) proposed the following thresholds for the Kappa statistic: less than 0 indicates no agreement, 0.01 to 0.20 indicates none to a slight agreement, 0.21 to 0.40

indicates fair agreement, 0.41 to 0.60 indicates moderate agreement, 0.61 to 0.80 indicates a substantial agreement, and 0.81 to 1.00 indicates nearly perfect agreement.

Chapter 3

Methodology

3.1 Introduction

In this chapter, the study describes 24 simulated classification scenarios generated to compare the performance of the LR model with machine learning models, namely, the ANN, WkNN, and the SVM. The analysis was carried out in R (R Core Team, 2021), using caret (Kuhn, 2021), kknn (Schliep and Hechenbichler, 2016), nnet (Venables and Ripley, 2002), NeuralNetTools (Beck, 2018), NeuralSens (Portela González et al., 2020), e1070 (Meyer et al., 2021), ModelMetrics (Hunt, 2020), dbplyr (Wickham et al., 2021), and dplyr (Wickham et al., 2021) packages. The code is available in Appendix B.

3.2 Data Generation and Analysis

This section outlines the data generation mechanism, fitted model performance, and findings.

3.2.1 Generation of Pseudo Code

The study generated data using simulations which are computer experiments that involve creating data by pseudo-random sampling from known probability distributions

(Morris et al., 2019). Different data-generating mechanisms were used to ensure coverage of different scenarios. This study targeted the predictive performance of the classification methods. The following steps were taken to simulate data:

1. Generate coefficients, $\beta_0, \beta_2, \ldots, \beta_j$
2. Generate the errors, ϵ
3. Generate predictors, $X_1, X_2, \ldots, X_j$
    3.1. $X_1, \ldots, X_s \sim$ were binned into categorical variables, and $X_{s+1}, \ldots, X_{j-s}$ are uniform random numbers, and $s < j$
4. $Y = e^{\boldsymbol{\beta}^\tau \mathbf{X} + \epsilon} / 1 + e^{\boldsymbol{\beta}^\tau \mathbf{X} + \epsilon}$

3.2.2 Binning Predictors

Simulations S1-S12 are different simulation scenarios generated to study the impact of binning predictors, the details are provided in Table 3.1. The first dataset included three predictors, x_1, x_2, and x_3 with a sample size of 1 000, and one response variable, y, with two classes. Predictors were generated from the random uniform distribution in the range [0, 1]. All predictors were continuous for S1 to S3; then, x_1 was binned into two classes in S4 to S6. Furthermore, from S7 to S9, x_1 and x_2 were binned into two classes. Then for S10 to S12, both x_1 and x_2 were binned but, in this case, into three classes. In the case of two classes, a continuous variable value was converted to the category 'A' if it was less than 0.5 and to 'B' otherwise. Each continuous variable was labeled as 'A' when it was less than 0.25, 'B' when it was between 0.25 and 0.75, and 'C' otherwise for two categorical variables with three classes. The coefficients were prespecified and fixed according to the relationship described in Table 3.1. These were adjusted to vary the probability of success (i.e $y = 1$) to introduce an imbalance in response classes. The response variable was generated from a random binomial distribution with the probability of success set to $Prob = 1/(1 + \exp(-(\mathbf{x}\boldsymbol{\beta} + \epsilon)))$, where ϵ is the random error generated from a random normal distribution.

Table 3.1: Binning predictors simulation description

Sim ID	Avg. Prob $(y = 1)$	Binned predictors	Classes of binned predictors	Relationship	Description
S1	$\pm$ 0.3	0	0	$y = -0.7 - x_1 - x_2 - x_3 + \epsilon$	No conversion
S2	$\pm$ 0.5	0	0	$y = 0.001 + 0.003x_1 + 0.001x_2 + 0.02x_3 + \epsilon$	No conversion
S3	$\pm$ 0.7	0	0	$y = 0.7+x_1+x_2+x_3+\epsilon$	No conversion
S4	$\pm$ 0.3	1	2	$y = -0.7 - x_1 - x_2 - x_3 + \epsilon$	x_1 is binned taking values A or B
S5	$\pm$ 0.5	1	2	$y = 0.001 + 0.003x_1 + 0.001x_2 + 0.02x_3 + \epsilon$	x_1 is binned taking values A or B
S6	$\pm$ 0.7	1	2	$y = 0.7+x_1+x_2+x_3+\epsilon$	x_1 is binned taking values A or B
S7	$\pm$ 0.3	2	2	$y = -0.7 - x_1 - x_2 - x_3 + \epsilon$	x_1 and x_2 are binned taking values A or B
S8	$\pm$ 0.5	2	2	$y = 0.001 + 0.003x_1 + 0.001x_2 + 0.02x_3 + \epsilon$	x_1 and x_2 are binned taking values A or B
S9	$\pm$ 0.7	2	2	$y = 0.7+x_1+x_2+x_3+\epsilon$	x_1 and x_2 are binned taking values A or B
S10	$\pm$ 0.3	2	3	$y = -0.7 - x_1 - x_2 - x_3 + \epsilon$	x_1 and x_2 are binned taking values A, B or C
S11	$\pm$ 0.5	2	3	$y = 0.001 + 0.003x_1 + 0.001x_2 + 0.02x_3 + \epsilon$	x_1 and x_2 are binned taking values A, B or C
S12	$\pm$ 0.7	2	3	$y = 0.7+x_1+x_2+x_3+\epsilon$	x_1 and x_2 are binned taking values A, B or C

3.2.3 Skewness

To study the effect of skewness in predictors, three numerical predictors were generated randomly from the log-normal distribution, and the models were applied to this

dataset while varying the probability of success and the number of repetitions as described above. Table 3.2 shows the process followed in skewing predictors.

Table 3.2: Skewed predictors simulation description

Sim ID	Avg. Prob $(y = 1)$	Relationship
S13	$\pm$ 0.3	$y = -0.7 - x_1 - x_2 - x_3 + \epsilon$
S14	$\pm$ 0.5	$y = 0.001 + 0.003x_1 + 0.001x_2 + 0.02x_3 + \epsilon$
S15	$\pm$ 0.7	$y = 0.7 + x_1 + x_2 + x_3 + \epsilon$

3.2.4 Data Dimensionality

To evaluate the impact of high-dimension on the model performance, the value of dimension (d) was set higher than the value of the sample size (n). Due to limited computer capabilities, the sample size was reduced to 100 for this scenario. The number of predictors was then set to 150 to introduce high-dimension. The intercepts were fixed at -0.7, 0.001, and 0.1 for the average probability of success of 0.3, 0.5, and 0.7 respectively. The coefficients were simulated from a random uniform distribution from the intervals $[-0.2, 0.2]$, $[0.001, 0.01]$, $[0.001, 0.03]$ to yield the average probability of success of 0.3, 0.5, and 0.7 respectively as shown in Table 3.3.

Table 3.3: High dimension simulation description

Sim ID	Avg. Prob $(y = 1)$	No. of predictors	Relationship
S16	$\pm$ 0.3	150	$y = \boldsymbol{\beta}^\tau \mathbf{X} + \epsilon, x_i = 1, 2, \ldots, 150, \beta_i \neq 0$
S17	$\pm$ 0.5	150	$y = \boldsymbol{\beta}^\tau \mathbf{X} + \epsilon, x_i = 1, 2, \ldots, 150, \beta_i \neq 0$
S18	$\pm$ 0.7	150	$y = \boldsymbol{\beta}^\tau \mathbf{X} + \epsilon, x_i = 1, 2, \ldots, 150, \beta_i \neq 0$

3.2.5 Predictor Selection

To identify predictor variables in a model that are important for classification, recursive feature elimination (RFE) (Guyon et al., 2002) was used. The model performance was

evaluated with and without these predictors in the model. The study generated a dataset with ten predictors ($y = \boldsymbol{\beta}^{\tau}\mathbf{X} + \epsilon, x_i = 1, 2, \ldots, 10, \beta_i \neq 0$), where the coefficients were prespecified to adjust the average probability of success to 0.3, 0.5, and 0.7 to investigate the issue of unbalanced response classes. The LR, ANN, SVM, and WkNN models were fitted to the different simulation examples described in Table 3.4, and after that, predictor selection was performed on each model using RFE with 50 repeated 10-fold cross-validation. The performance of the models before and after predictor selection were compared.

Table 3.4: Predictor selection simulation description

Sim ID	Avg. Prob $(y = 1)$	No. of predictors	Relationship
S25	$\pm$ 0.3	10	$-0.7 - x_1 - x_2 - x_3 + 0.15x_4 + 0.2x_5 - 0.03x_6 - 0.1x_7 + 0.01x_8 + 0.5x_9 + 0.1x_{10} + \epsilon$
S26	$\pm$ 0.5	10	$0.001 + 0.003x_1 + 0.001x_2 + 0.02x_3 + 0.03x_4 + 0.01x_5 + 0.011x_6 + 0.015x_7 + 0.05x_8 + 0.07x_9 + 0.01x_{10} + \epsilon$
S27	$\pm$ 0.7	10	$0.7 + x_1 + x_2 + x_3 + 0.2x_4 + 0.05x_5 - 0.05x_6 + 0.02x_7 + 0.03x_8 + 0.01x_9 - 0.01x_{10} + \epsilon$

3.2.6 Methods

The k-fold cross-validation resampling method was used to train and test the models' performance on different iterations. The value of k varied between 5 and 10 but 10 gave better results, and therefore the commonly used 10-fold cross-validation was used throughout the analysis.

A nnet function in R was used to train a feedforward neural network with one hidden layer using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimisation to find internal weight constants. A size 3 neural network with a weight decay parameter of 0.0001

gave the best accuracy and Kappa statistic, and this is the architecture adopted in the analysis. LR used both the logit and probit link functions. In the WkNN model, k was varied from 1 to 10 to determine its optimal value; a k of size 9 with Minkowski distance and an optimal kernel yielded better results. And lastly, a linear SVM was used.

Chapter 4

Results and Analysis

4.1 Introduction

This chapter describes the data used and the results of the fitted LR (both with logit and probit links), 2-layer ANN, linear SVM, and WkNN models. Accuracy, Kappa statistic, AUC, sensitivity, and specificity were used as metrics to measure these models' performance. Since the models were run repeatedly, with repeats varied from 50, 100, and 500, the mean, median, and standard error across repetitions were calculated. This study reports the mean of the metrics, as the results were consistent in each simulation with very low variation. However, one example that reports the median and standard error results is displayed in Appendix A.1.

4.2 Data Description

As described in Section 3.2, various simulations were generated to fit the models. The data summary for all numeric predictors, one binned predictor, two binned predictors, and when two predictors were binned into three classes is shown in the Tables 4.1, 4.2, 4.3, and 4.4 below. In addition, the probability of success ($y = 1$) was set to an average of 0.3 in this case, resulting in an imbalance in the response classes. Furthermore, the distribution of the data is displayed in Figures 4.1a, 4.1b, 4.1c, and 4.1d.

As seen in the histograms, the predictors were generated from a random uniform distribution with a [0,1] range. This is one of many simulations carried out as part of this research. The case of skewed predictors revealed a skewed distribution. Additionally, in the case of high-dimensional data, the number of predictors was set to 150 and the sample size was reduced to 100 to study the effect of high-dimensional data on model performance. All predictors were numeric and as a result, the distribution resembled that of the all-numerical case shown here.

Table 4.1: Numeric predictors data summary

x_1	x_2	x_3	y
Min. : 0.0007932	Min. : 0.001663	Min. : 0.0001093	failure: 853
1st Qu. : 0.2467932	1st Qu. : 0.244198	1st Qu. : 0.2308714	success: 147
Median : 0.4862225	Median : 0.498867	Median : 0.4994472	
Mean : 0.4993531	Mean : 0.503871	Mean : 0.5010679	
3rd Qu : 0.7550995	3rd Qu : 0.766742	3rd Qu : 0.7487922	
Max : 0.9994952	Max : 0.999192	Max : 0.9982291	

Table 4.2: One binned predictor data summary

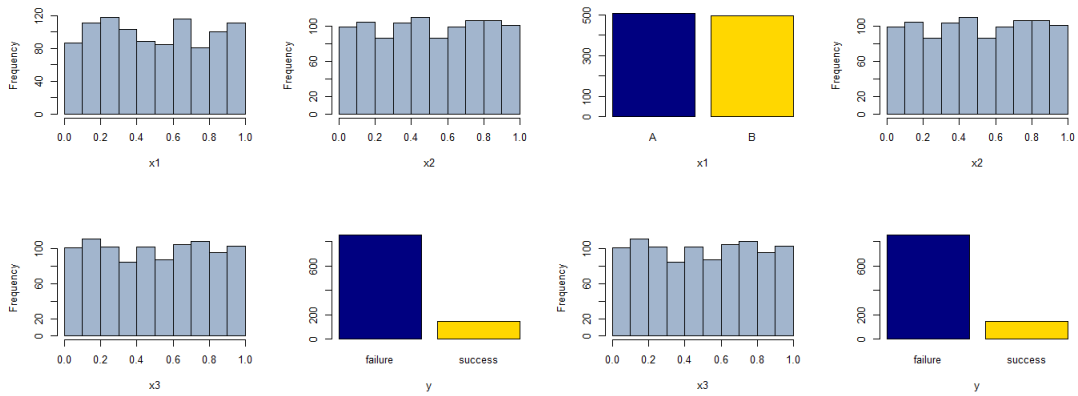
x_1	x_2	x_3	y
A: 507	Min. : 0.001663	Min. : 0.0001093	failure:853
B: 493	1st Qu. : 0.244198	1st Qu. : 0.2308714	success:147
	Median : 0.498867	Median : 0.4994472	
	Mean : 0.503871	Mean : 0.5010679	
	3rd Qu : 0.766742	3rd Qu : 0.7487922	
	Max : 0.999192	Max : 0.9982291	

Table 4.3: Two binned predictors data summary

x_1	x_2	x_3	y
A: 507	A: 502	Min. : 0.0001093	failure: 853
B: 493	B: 498	1st Qu. : 0.2308714	success: 147
		Median : 0.4994472	
		Mean : 0.5010679	
		3rd Qu : 0.7487922	
		Max : 0.9982291	

Table 4.4: Two binned predictors (three classes) data summary

x_1	x_2	x_3	y
A: 253	A: 254	Min. : 0.0001093	failure: 853
B: 254	B: 248	1st Qu. : 0.2308714	success: 147
C: 493	C: 498	Median : 0.4994472	
		Mean : 0.5010679	
		3rd Qu : 0.7487922	
		Max : 0.9982291	



(a) Numeric predictors

(b) x_1 binned into 2 classes

(c) x_1 and x_2 binned into 2 classes

(d) x_1 and x_2 binned into 3 classes

Figure 4.1: Binning predictors data distribution

4.3 The Fitted Models Results

4.3.1 Binning Predictors Results

The LR, ANN, SVM, and the WkNN were fitted to simulations S1 to S3 with three continuous predictors, and the accuracy, Kappa statistic, AUC, sensitivity, and specificity results are shown in Tables 4.5, 4.9, and 4.13. Thereafter, one predictor was binned into two classes, and Tables 4.6, 4.10, and 4.14 display the results. Furthermore, two predictors were binned into two classes, with results displayed in Tables 4.7, 4.11, and 4.15. And lastly, two predictors were binned but in this case into three classes, and models results are shown in Tables 4.8, 4.12, and 4.16. The impact of binning predictors on model performance was also studied while considering balanced and unbalanced response classes.

4.3.1.1 Accuracy: Binning Predictors

Table 4.5: Accuracy for LR, ANN, SVM, and WkNN models considering unbalanced classes with continuous predictors

Sim ID	Reps	Avg. Prob $(y = 1)$	Accuracy				
			Logit	Probit	ANN	SVM	WkNN
S1	50	$\pm$ 0.3	0.85	0.85	0.85	0.85	0.83
	100		0.85	0.85	0.85	0.85	0.83
	500		0.85	0.85	0.85	0.85	0.83
S2	50	$\pm$ 0.5	0.51	0.51	0.51	0.53	0.47
	100		0.51	0.51	0.51	0.53	0.47
	500		0.51	0.51	0.51	0.53	0.47
S3	50	$\pm$ 0.7	0.85	0.85	0.84	0.85	0.83
	100		0.85	0.85	0.84	0.85	0.83
	500		0.85	0.85	0.84	0.85	0.83

The models appeared to overestimate accuracy when response classes were unbalanced, as shown in the table above. In the presence of unbalance, as a result of as-

signing observations to the majority class, the accuracies were generally higher. LR, both with logit and probit link functions, and ANN performed the same when the response classes were balanced, achieving an accuracy of 51%, which is comparable to a guess. This contradicts the findings of Kim (2010), which compared the LR and ANN for a binary response and three numerical predictors. Kim (2010) discovered that the LR performed better than the ANN under these conditions; however, in this study, the performance of these two models was identical. The WkNN performed poorly with the lowest accuracy of 47%, whereas the SVM provided the highest accuracy of 53%, although still low. The models were consistent across all repetitions.

Table 4.6: Accuracy for LR, ANN, SVM, and WkNN models considering unbalanced classes with two continuous predictors and one binned predictor (two classes)

Sim ID	Reps	Avg. Prob $(y = 1)$	Accuracy				
			Logit	Probit	ANN	SVM	WkNN
S4	50	$\pm$ 0.3	0.85	0.85	0.85	0.85	0.83
	100		0.85	0.85	0.85	0.85	0.83
	500		0.85	0.85	0.85	0.85	0.83
S5	50	$\pm$ 0.5	0.51	0.51	0.51	0.53	0.50
	100		0.51	0.51	0.51	0.53	0.50
	500		0.51	0.51	0.51	0.53	0.50
S6	50	$\pm$ 0.7	0.85	0.85	0.85	0.85	0.83
	100		0.85	0.85	0.85	0.85	0.83
	500		0.85	0.85	0.85	0.85	0.83

When one predictor was binned into two classes, the models' accuracies remained relatively unchanged from what it was with numeric predictors, with the exception of the WkNN, whose accuracy increased from 47% to 50%, despite still performing poorly. Once the probability was set to 0.7, the ANN model accuracy also increased from 84% to 85%. LR, ANN, and SVM all performed identically with an accuracy of 85% for unbalanced classes, followed by WkNN with an accuracy of 83%. However,

when classes were balanced, the SVM outperformed the other models with an accuracy of 53%, followed by the LR and ANN models with 51%, and the WkNN with 50%. In contrast to the results of Kim (2010), which demonstrated that the ANN outperformed the LR under these conditions, the LR produced identical results to the ANN. All models performed poorly for balanced classes and overestimated for unbalanced ones.

Table 4.7: Accuracy for LR, ANN, SVM, and WkNN models considering unbalanced classes with one continuous predictor and two binned predictors (two classes)

Sim ID	Reps	Avg. Prob $(y = 1)$	Accuracy				
			Logit	Probit	ANN	SVM	WkNN
S7	50	$\pm$ 0.3	0.85	0.85	0.85	0.85	0.84
	100		0.85	0.85	0.85	0.85	0.84
	500		0.85	0.85	0.85	0.85	0.84
S8	50	$\pm$ 0.5	0.52	0.52	0.50	0.53	0.46
	100		0.52	0.52	0.51	0.53	0.46
	500		0.52	0.52	0.51	0.53	0.46
S9	50	$\pm$ 0.7	0.85	0.85	0.85	0.85	0.83
	100		0.85	0.85	0.85	0.85	0.83
	500		0.85	0.85	0.85	0.85	0.83

When the probability was set to 0.3, the WkNN improved its accuracy from 83% to 84%, while it dropped from 48% to 46% for balanced classes. The WkNN model appeared to overestimate the accuracy more when two predictors were binned into two classes, although this was not a significant change. However, when classes were balanced, the LR model's accuracy improved from 51% to 52% due to the binning of two predictors.

Table 4.8: Accuracy for LR, ANN, SVM, and WkNN models considering unbalanced classes with one continuous predictor and two binned predictors (three classes)

Sim ID	Reps	Avg. Prob $(y = 1)$	Accuracy				
			Logit	Probit	ANN	SVM	WkNN
S10	50	$\pm$ 0.3	0.85	0.85	0.85	0.85	0.83
	100		0.85	0.85	0.85	0.85	0.83
	500		0.85	0.85	0.85	0.85	0.83
S11	50	$\pm$ 0.5	0.51	0.50	0.50	0.53	0.50
	100		0.51	0.50	0.50	0.53	0.50
	500		0.50	0.50	0.50	0.53	0.50
S12	50	$\pm$ 0.7	0.85	0.85	0.85	0.85	0.82
	100		0.85	0.85	0.85	0.85	0.82
	500		0.85	0.85	0.85	0.85	0.82

In the case of balanced classes, the accuracy of the LR and ANN decreased from 51% to 50%, whereas the accuracy of the WkNN increased from 47% to 50%, indicating that these three models classified observations only by chance. The SVM maintained consistency of 53% accuracy regardless of binning predictors, outperforming competing models. Kim (2010) further discovered that for balanced response classes, one numerical predictor, and two binned predictors with three classes, the ANN performed better than the LR which was not the case in this study since the LR and ANN yielded the same results.

Summary

Overall, the models overestimated the accuracy when classes were unbalanced, which was reflected in the poor performance when classes were balanced. The LR and ANN performed similarly, the SVM consistently outperformed the competing models, and the WkNN performed worse with the lowest accuracy.

4.3.1.2 Kappa Statistic: Binning Predictors Results

Table 4.9: Kappa statistic for LR, ANN, SVM, and WkNN models considering unbalanced classes with continuous predictors

Sim ID	Reps	Avg. Prob $(y = 1)$	Kappa statistic				
			Logit	Probit	ANN	SVM	WkNN
S1	50	$\pm$ 0.3	0.00	0.00	0.00	0.00	0.04
	100		0.00	0.00	0.00	0.00	0.04
	500		0.00	0.00	0.00	0.00	0.04
S2	50	$\pm$ 0.5	-0.03	-0.03	-0.01	0.00	-0.07
	100		-0.02	-0.03	-0.01	0.00	-0.07
	500		-0.03	-0.02	-0.01	0.00	-0.07
S3	50	$\pm$ 0.7	0.00	0.00	0.00	0.00	-0.02
	100		0.00	0.00	0.00	0.00	-0.01
	500		0.00	0.00	0.00	0.00	-0.02

Overall, all models performed poorly in separating classes; this is evident in extremely low Kappa statistic values shown in the preceding table, indicating no agreement between the predicted and the observed classes. However, the WkNN performed better than the other models when the average probability of success was approximately 0.3, with 0.04 Kappa statistic though still indicating none to a slight agreement. Although the WkNN appeared to provide lower accuracy compared to the other models, it performed better in separating classes than the competing models with higher Kappa values, as shown in Table 4.9, particularly when the probability was set to 0.3.

Table 4.10: Kappa statistic for LR, ANN, SVM, and WkNN models considering unbalanced classes with two continuous predictors and one binned predictor (two classes)

Sim ID	Reps	Avg. Prob $(y = 1)$	Kappa statistic				
			Logit	Probit	ANN	SVM	WkNN
S4	50	$\pm$ 0.3	0.00	0.00	0.00	0.00	-0.02
	100		0.00	0.00	0.00	0.00	-0.02
	500		0.00	0.00	0.00	0.00	-0.02
S5	50	$\pm$ 0.5	-0.03	-0.03	-0.01	0.00	-0.01
	100		-0.03	-0.03	0.00	0.00	-0.01
	500		-0.03	-0.03	0.00	0.00	-0.01
S6	50	$\pm$ 0.7	0.00	0.00	0.00	0.00	-0.01
	100		0.00	0.00	0.00	0.00	0.00
	500		0.00	0.00	0.00	0.00	-0.01

The models continued to perform poorly after one predictor was binned into two classes with all Kappa statistic values indicating no agreement between predicted and observed classes. While the WkNN performed better (for unbalanced classes with a success probability of 0.3) when all predictors were numeric, its performance also declined when one predictor was binned.

Table 4.11: Kappa statistic for LR, ANN, SVM, and WkNN models considering unbalanced classes with one continuous predictor and two binned predictors (two classes)

Sim ID	Reps	Avg. Prob $(y = 1)$	Kappa statistic				
			Logit	Probit	ANN	SVM	WkNN
S7	50	$\pm$ 0.3	0.00	0.00	0.00	0.00	0.03
	100		0.00	0.00	0.00	0.00	0.03
	500		0.00	0.00	0.00	0.00	0.03
S8	50	$\pm$ 0.5	-0.02	-0.02	-0.02	0.00	-0.08
	100		-0.02	-0.02	-0.02	0.00	-0.08
	500		-0.02	-0.02	-0.02	0.00	-0.08
S9	50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.02
	100		0.00	0.00	0.00	0.00	0.01
	500		0.00	0.00	0.00	0.00	0.01

The WkNN performed better based on Kappa statistic values for unbalanced classes. For a success probability of 0.7, its Kappa values improved from -0.02 with numeric predictors to 0.01 when two predictors were binned into two classes. However, there was a slight decline in the Kappa statistic for this model for a success probability of 0.3. Although this model's Kappa statistic still showed none to a slight agreement between the predicted and observed classes, it performed better compared to the other models that showed no agreement.

Table 4.12: Kappa statistic for LR, ANN, SVM, and WkNN models considering unbalanced classes with one continuous predictor and two binned predictors (three classes)

Sim ID	Reps	Avg. Prob $(y = 1)$	Kappa statistic				
			Logit	Probit	ANN	SVM	WkNN
S10	50	$\pm$ 0.3	0.00	0.00	0.00	0.00	0.01
	100		0.00	0.00	0.00	0.00	0.01
	500		0.00	0.00	0.00	0.00	0.01
S11	50	$\pm$ 0.5	-0.04	-0.04	-0.01	0.00	-0.01
	100		-0.04	-0.04	-0.02	0.00	-0.01
	500		-0.04	-0.04	-0.01	0.00	-0.01
S12	50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.00
	100		0.00	0.00	0.00	0.00	0.00
	500		0.00	0.00	0.00	0.00	0.00

Based on 4.12, the models' Kappa statistics remained poor though the WkNN performed better than the other models with a 0.3 probability of success. However, its performance declined from 0.04 with all numeric predictors to 0.01 with two predictors binned into three classes.

Summary

All models performed poorly in separating classes, yielding extremely low Kappa statistics. However, the WkNN outperformed the competing models based on the Kappa statistic when classes were unbalanced, particularly with a 0.7 success probability. It appeared that for unbalanced classes, binning two predictors into two classes improved the performance of the WkNN model, although with a slight tradeoff on the Kappa statistic for a 0.3 success probability.

4.3.1.3 AUC: Binning Predictors

This section examines the effect of binning predictors on the performance of the LR, ANN, SVM, and WkKNN by displaying and discussing the AUC results of the models fitted to S1 through S12.

Table 4.13: AUC for LR, ANN, SVM, and WkNN models considering unbalanced classes with continuous predictors

Sim ID	Reps	Avg. Prob $(y = 1)$	AUC				
			Logit	Probit	ANN	SVM	WkNN
S1	50	$\pm$ 0.3	0.61	0.61	0.58	0.49	0.56
	100		0.62	0.61	0.58	0.50	0.55
	500		0.61	0.61	0.58	0.50	0.55
S2	50	$\pm$ 0.5	0.47	0.47	0.49	0.50	0.47
	100		0.47	0.47	0.50	0.50	0.47
	500		0.47	0.47	0.50	0.50	0.47
S3	50	$\pm$ 0.7	0.51	0.51	0.50	0.50	0.49
	100		0.51	0.51	0.50	0.50	0.49
	500		0.51	0.51	0.50	0.50	0.49

In contrast to the accuracy results in Tables 4.5 to 4.8, which were misleading in terms of how well the models performed, particularly for unbalanced classes, it is evident from the AUC results in the table above that the models poorly classified the classes. The LR performed better compared to the other models for unbalanced classes, whereas the ANN and SVM performed better for balanced classes, although this was purely by chance.

Table 4.14: AUC for LR, ANN, SVM, and WkNN models considering unbalanced classes with two continuous predictors and one binned predictor (two classes)

Sim ID	Reps	Avg. Prob $(y = 1)$	AUC				
			Logit	Probit	ANN	SVM	WkNN
S4	50	$\pm$ 0.3	0.61	0.61	0.58	0.51	0.53
	100		0.61	0.61	0.57	0.50	0.53
	500		0.61	0.61	0.57	0.50	0.53
S5	50	$\pm$ 0.5	0.48	0.48	0.50	0.50	0.50
	100		0.48	0.48	0.50	0.51	0.50
	500		0.48	0.48	0.50	0.50	0.50
S6	50	$\pm$ 0.7	0.53	0.53	0.52	0.50	0.54
	100		0.53	0.53	0.52	0.50	0.53
	500		0.53	0.53	0.52	0.50	0.53

The performance of the LR, and SVM remained unchanged after one predictor was binned, while that of the ANN and WkNN saw a slight decline from 58% to 57% and 55% to 53% respectively. However, for a 0.7 success probability, the AUC of the LR, ANN, and WkNN all improved from 51% to 53%, 50% to 52%, and 49% to 53% respectively, while the SVM's AUC remained the same. In the case of balanced classes, the LR, and WkNN improved from 47% to 48% and 47% to 50% respectively. Despite this, these models' performance was poor.

Table 4.15: AUC for LR, ANN, SVM, and WkNN models considering unbalanced classes with one continuous predictor and two binned predictors (two classes)

Sim ID	Reps	Avg. Prob $(y = 1)$	AUC				
			Logit	Probit	ANN	SVM	WkNN
S7	50	$\pm$ 0.3	0.61	0.61	0.57	0.50	0.50
	100		0.61	0.61	0.57	0.50	0.50
	500		0.63	0.63	0.59	0.50	0.50
S8	50	$\pm$ 0.5	0.48	0.48	0.49	0.50	0.47
	100		0.48	0.48	0.49	0.50	0.46
	500		0.48	0.48	0.49	0.50	0.46
S9	50	$\pm$ 0.7	0.53	0.53	0.52	0.50	0.54
	100		0.53	0.53	0.52	0.50	0.54
	500		0.53	0.53	0.52	0.50	0.54

The performance of the LR improved when two predictors were binned into two classes for both balanced and unbalanced classes. The AUC of the ANN model improved for unbalanced classes but declined when classes were balanced. The WkNN only improved with a higher probability of success (0.7). The SVM remained consistent and was not affected by binning predictors.

Table 4.16: AUC for LR, ANN, SVM, and WkNN models considering unbalanced classes with one continuous predictor and two binned predictors (three classes)

Sim ID	Reps	Avg. Prob $(y = 1)$	AUC				
			Logit	Probit	ANN	SVM	WkNN
S10	50	$\pm$ 0.3	0.61	0.61	0.58	0.51	0.48
	100		0.61	0.61	0.58	0.50	0.48
	500		0.61	0.61	0.58	0.50	0.48
S11	50	$\pm$ 0.5	0.45	0.45	0.49	0.50	0.50
	100		0.45	0.45	0.49	0.51	0.49
	500		0.45	0.45	0.49	0.51	0.50
S12	50	$\pm$ 0.7	0.55	0.55	0.54	0.50	0.55
	100		0.55	0.55	0.54	0.50	0.55
	500		0.53	0.53	0.52	0.50	0.54

Given balanced response classes, binning two predictors into three classes declined the AUC of the LR and ANN but improved that of the SVM and WkNN. The AUC of the WkNN declined when the probability of success was set to 0.3 but improved as the probability increased to 0.7. The SVM maintained the same performance for unbalanced classes.

Summary

Based on the AUC results, the classification performance of the models was poor. Nonetheless, it appeared that binning predictors favored the LR, given that predictors were binned into two classes. The performance of the SVM only slightly improved when the response classes were balanced and two predictors were binned into three classes. The WkNN performed better with all numeric predictors when the probability of success was set to 0.3 when compared to the other WkNN models; however, for balanced classes, the WkNN models with one binned predictor and two predictors binned into three classes gave better results, and when the probability of success was set to 0.7, the WkNN models with two predictors binned into two or three classes performed better. For unbalanced classes (probability set to 0.3), the ANN models with

two binned predictors outperformed other ANN models, whereas, for balanced classes, the ANN models with all numeric predictors and one binned predictor performed better. In addition, when the probability of success was increased to 0.7, all ANN models with binned predictors outperformed models with only numeric predictors.

4.3.1.4 Sensitivity and Specificity: Binning Predictors

Table 4.17: Sensitivity and specificity for LR, ANN, SVM, and WkNN models considering unbalanced classes with continuous predictors

Sim ID	Reps	Pr($y = 1$)	Sensitivity					Specificity				
			Logit	Probit	ANN	SVM	WkNN	Logit	Probit	ANN	SVM	WkNN
S1	50	$\pm$ 0.3	1.00	1.00	1.00	1.00	0.97	0.00	0.00	0.00	0.00	0.06
	100		1.00	1.00	1.00	1.00	0.97	0.00	0.00	0.00	0.00	0.06
	500		1.00	1.00	1.00	1.00	0.97	0.00	0.00	0.00	0.00	0.06
S2	50	$\pm$ 0.5	0.04	0.04	0.26	0.00	0.43	0.94	0.94	0.73	1.00	0.50
	100		0.03	0.04	0.24	0.00	0.43	0.94	0.94	0.75	1.00	0.50
	500		0.03	0.03	0.25	0.00	0.43	0.94	0.94	0.74	1.00	0.50
S3	50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.01	1.00	1.00	1.00	1.00	0.98
	100		0.00	0.00	0.00	0.00	0.01	1.00	1.00	1.00	1.00	0.98
	500		0.00	0.00	0.00	0.00	0.01	1.00	1.00	1.00	1.00	0.98

When response classes were unbalanced and the probability of success was set to 0.3, the models returned high sensitivity but low specificity, indicating that the classifiers were able to correctly identify "failure $(y = 0)$" but not "success $(y = 1)$". In this case, the LR, ANN, and SVM returned 100% sensitivity and 0% specificity, whereas the WkNN yielded 97% sensitivity and 6% specificity. In the unbalanced classes in which the success probability was 0.7, the opposite was true, the results showed high specificity and low sensitivity. The results of the unbalanced datasets were attributed to majority class bias. For balanced classes, classifiers continued to have, in general, higher specificity than sensitivity. The ANN appeared to perform better, correctly classifying 74% of instances as "success" and 25% as "failure." The WkNN model was comparable to a random guess.

Table 4.18: Sensitivity and specificity for LR, ANN, SVM, and WkNN models considering unbalanced classes with two continuous predictors and one binned predictor (two classes)

Sim ID	Reps	Pr($y = 1$)	Sensitivity					Specificity				
			Logit	Probit	ANN	SVM	WkNN	Logit	Probit	ANN	SVM	WkNN
S4	50	$\pm$ 0.3	1.00	1.00	1.00	1.00	0.97	0.00	0.00	0.00	0.00	0.01
	100		1.00	1.00	1.00	1.00	0.97	0.00	0.00	0.00	0.00	0.01
	500		1.00	1.00	1.00	1.00	0.97	0.00	0.00	0.00	0.00	0.01
S5	50	$\pm$ 0.5	0.05	0.05	0.30	0.00	0.46	0.92	0.92	0.70	1.00	0.53
	100		0.05	0.05	0.30	0.00	0.46	0.92	0.92	0.69	1.00	0.54
	500		0.05	0.05	0.30	0.00	0.46	0.92	0.92	0.70	1.00	0.53
S6	50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.02	1.00	1.00	1.00	1.00	0.97
	100		0.00	0.00	0.00	0.00	0.02	1.00	1.00	1.00	1.00	0.97
	500		0.00	0.00	0.00	0.00	0.02	1.00	1.00	1.00	1.00	0.97

The specificity of the WkNN declined from 6% to 1% when the success probability was set to 0.3, from 98% to 97% when it was set to 0.7, and improved sensitivity from 43% to 46% for balanced classes and from 1% to 2% when the success probability was set to 0.7. For balanced classes, the sensitivity of the LR and ANN increased from 3% to 5% and 25% to 30%, respectively, while specificity decreased from 94% to 92% and 74% to 70%, respectively. The SVM produced consistent results for both balanced and unbalanced response classes.

Table 4.19: Sensitivity and specificity for LR, ANN, SVM, and WkNN models considering unbalanced classes with one continuous predictor and two binned predictors (two classes)

Sim ID	Reps	Pr($y = 1$)	Sensitivity					Specificity				
			Logit	Probit	ANN	SVM	WkNN	Logit	Probit	ANN	SVM	WkNN
S7	50	$\pm$ 0.3	1.00	1.00	1.00	1.00	0.98	0.00	0.00	0.00	0.00	0.04
	100		1.00	1.00	1.00	1.00	0.98	0.00	0.00	0.00	0.00	0.04
	500		1.00	1.00	1.00	1.00	0.98	0.00	0.00	0.00	0.00	0.04
S8	50	$\pm$ 0.5	0.04	0.04	0.25	0.00	0.43	0.94	0.94	0.73	0.99	0.50
	100		0.04	0.04	0.26	0.01	0.42	0.94	0.94	0.72	0.99	0.50
	500		0.04	0.04	0.26	0.01	0.43	0.94	0.94	0.73	0.99	0.50
S9	50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.04	1.00	1.00	1.00	1.00	0.97
	100		0.00	0.00	0.00	0.00	0.04	1.00	1.00	1.00	1.00	0.97
	500		0.00	0.00	0.00	0.00	0.04	1.00	1.00	1.00	1.00	0.97

The WkNN's sensitivity improved from 97% to 98% and specificity declined from 6% to 4% when the probability was set to 0.3. Upon increasing the probability to 0.7, the sensitivity improved from 1% to 4% while specificity declined from 98% to 97%. For the balanced dataset, the results were the same as the ones with all numeric predictors. The LR, ANN, and SVM models' sensitivity improved from 3% to 4%, 25% to 26%, and 0% to 1% respectively for a balanced dataset while the specificity stayed the same for the LR, declined from 74% to 73% for the ANN, and from 100% to 99% for the SVM.

Table 4.20: Sensitivity and specificity for LR, ANN, SVM, and WkNN models considering unbalanced classes with one continuous predictor and two binned predictors (three classes)

Sim ID	Reps	Pr($y = 1$)	Sensitivity					Specificity				
			Logit	Probit	ANN	SVM	WkNN	Logit	Probit	ANN	SVM	WkNN
S10	50	$\pm$ 0.3	1.00	1.00	1.00	1.00	0.97	0.00	0.00	0.01	0.00	0.04
	100		1.00	1.00	1.00	1.00	0.97	0.00	0.00	0.01	0.00	0.04
	500		1.00	1.00	1.00	1.00	0.97	0.00	0.00	0.01	0.00	0.04
S11	50	$\pm$ 0.5	0.05	0.05	0.37	0.01	0.44	0.91	0.91	0.61	0.99	0.55
	100		0.06	0.05	0.39	0.01	0.44	0.90	0.91	0.60	0.99	0.55
	500		0.05	0.05	0.38	0.01	0.44	0.91	0.90	0.60	0.99	0.55
S12	50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.03	1.00	1.00	1.00	1.00	0.97
	100		0.00	0.00	0.00	0.00	0.03	1.00	1.00	1.00	1.00	0.97
	500		0.00	0.00	0.00	0.00	0.04	1.00	1.00	1.00	1.00	0.97

For a success probability of 0.3, when two predictors were binned into three classes, the sensitivity of all models remained unchanged. While the specificity of the LR and SVM also remained unchanged, the ANN's specificity improved from 0% to 1%, and the WkNN decreased from 6% to 4%. When the response classes were balanced, all models' sensitivity increased and their specificity decreased, except for the WkNN, whose specificity increased from 50% to 55%. The results of all other models remained unchanged, whereas the WkNN model's specificity increased from 1% to 4% when the probability of success was set to 0.7%, and its specificity decreased from 98% to 97%.

Summary

Overall, the models lacked a balanced combination of sensitivity and specificity. When the success probability was 0.3, models were more sensitive to "failure," whereas when it was 0.7, models became more specific. In the case of balanced classes, models still attained a greater degree of specificity. Once one predictor was binned into two classes, the LR performed better than other LR models. The ANN performed better when two predictors were binned into three classes. SVM models with two binned

predictors in a balanced dataset performed better than the other SVM models, and the WkNN performed better with all numeric predictors when the probability of success was 0.7. For balanced response classes, the WkNN models with one binned predictor or two performed better, and when the probability was increased to 0.7, the WkNN models with two binned predictors performed better.

4.3.2 Skewed Predictors Results

This section investigates how skewed predictors affect model performance.

4.3.2.1 Accuracy: Skewed Predictors

The accuracy of models fitted to datasets with skewed predictors was evaluated, and the results are discussed below.

Table 4.21: Accuracy: unbalanced classes and skewed predictors

Sim ID	Reps	Avg. Prob $(y = 1)$	Accuracy				
			Logit	Probit	ANN	SVM	WkNN
S13	50	$\pm$ 0.3	0.96	0.96	0.96	0.96	0.96
	100		0.96	0.96	0.96	0.96	0.96
	500		0.96	0.96	0.96	0.96	0.96
S14	50	$\pm$ 0.5	0.51	0.51	0.51	0.52	0.50
	100		0.51	0.51	0.51	0.52	0.51
	500		0.51	0.51	0.51	0.52	0.51
S14	50	$\pm$ 0.7	0.97	0.97	0.97	0.97	0.96
	100		0.97	0.97	0.97	0.97	0.96
	500		0.97	0.97	0.97	0.97	0.96

All models performed comparably for the dataset with skewed predictors. In the case of unbalanced classes (probability set to 0.3), the models gave identical results with a 96% accuracy, and 97% when the probability was set to 0.7 with an exception of thE WkNN that maintained a 96% accuracy. The models were clearly biased toward the dominant class, as they performed poorly when response classes were balanced.

The SVM had a slightly higher accuracy of 52% compared to 51% for the rest of the models.

4.3.2.2 Kappa Statistic: Skewed Predictors

Using the Kappa statistic, the performance of the models fitted to the skewed predictors' datasets is discussed below.

Table 4.22: Kappa statistic: unbalanced classes and skewed predictors

Sim ID	Reps	Avg. Prob $(y = 1)$	Kappa statistic				
			Logit	Probit	ANN	SVM	WkNN
S13	50	$\pm$ 0.3	0.00	0.00	0.00	0.00	0.03
	100		0.00	0.00	0.00	0.00	0.03
	500		0.00	0.00	0.00	0.00	0.03
S14	50	$\pm$ 0.5	0.00	0.00	0.02	0.00	0.01
	100		0.00	0.00	0.01	0.00	0.01
	500		0.00	0.00	0.01	0.00	0.01
S15	50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.00
	100		0.00	0.00	0.00	0.00	0.00
	500		0.00	0.00	0.00	0.00	0.00

In general, none of the models were able to distinguish between the classes, as indicated by the extremely low Kappa statistic values. When the classes were balanced, the ANN and WkNN performed identically, outperforming competing models with Kappa statistics of 0.01, which still indicated a lack of agreement to a slight agreement between predicted and observed classes. The WkNN also outperformed the competition for unbalanced classes when the probability was set to 0.3 with a Kappa statistic of 0.03.

4.3.2.3 AUC: Skewed Predictors

Using the AUC, the performance of the models fitted to the skewed predictors' datasets is discussed below.

Table 4.23: AUC: unbalanced classes and skewed predictors

Sim ID	Reps	Avg. Prob $(y = 1)$	AUC				
			Logit	Probit	ANN	SVM	WkNN
S13	50	$\pm$ 0.3	0.84	0.84	0.80	0.55	0.65
	100		0.84	0.84	0.80	0.55	0.65
	500		0.84	0.84	0.80	0.55	0.65
S14	50	$\pm$ 0.5	0.52	0.52	0.51	0.49	0.50
	100		0.52	0.52	0.51	0.49	0.51
	500		0.52	0.52	0.51	0.49	0.51
S15	50	$\pm$ 0.7	0.82	0.82	0.73	0.70	0.59
	100		0.82	0.83	0.73	0.69	0.59
	500		0.82	0.82	0.72	0.70	0.59

When the classes were unbalanced, with a 0.3 probability of success, the LR performed the best with an AUC of 84%, followed by the ANN with 80%, WkNN with 66%, and the SVM with 55%. After balancing the classes, the LR continued to outperform the other models, in this case with a lower AUC of 52%, followed by the ANN and WkNN with 51% AUC, and finally the SVM with 49% AUC. Once the probability was increased to 0.7, the LR still outperformed the other models with an AUC of 82%, followed by the ANN with 72%, SVM with 70%, and finally the WkNN with 59%. Based on AUC results for both balanced and unbalanced datasets, the LR performed better than the competing models in separating classes.

4.3.2.4 Sensitivity and Specificity: Skewed Predictors

Using sensitivity and specificity, the performance of the models fitted to the skewed predictors' datasets is discussed below.

Table 4.24: Sensitivity and specificity: unbalanced classes and skewed predictors

Sim ID	Reps	Pr($y = 1$)	Sensitivity					Specificity				
			Logit	Probit	ANN	SVM	WkNN	Logit	Probit	ANN	SVM	WkNN
S13	50	$\pm$ 0.3	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.02
	100		1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.02
	500		1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.02
S14	50	$\pm$ 0.5	0.32	0.32	0.36	0.00	0.49	0.68	0.68	0.65	1.00	0.52
	100		0.32	0.32	0.37	0.00	0.50	0.68	0.68	0.64	1.00	0.52
	500		0.32	0.32	0.36	0.00	0.50	0.68	0.68	0.65	1.00	0.52
S15	50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00
	100		0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00
	500		0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00

The models maintained a high level of sensitivity (100%) for unbalanced classes with a probability of 0.3 despite having a low level of specificity. When the probability was raised to 0.7, the opposite occurred. On the balanced dataset, the LR had the highest specificity of 68% and 32% sensitivity, indicating that this model correctly classified 68% of observations as successes but only 32% as failures. The ANN followed with a specificity of 65% and a sensitivity of 36%. With a sensitivity of 0% and a specificity of 100%, the SVM failed to correctly classify failure but correctly identified success. The WkNN balanced sensitivity and specificity, but this model was only as accurate as a guess.

Summary

Although the models' performance was comparable across metrics, when the probability of success was set to 0.3, the WkNN performed better, particularly on the Kappa

statistic and specificity. For balanced classes, the WkNN and ANN models outperformed the other models, particularly in terms of the Kappa statistic, AUC, and sensitivity. These models did not always provide the best results in these metrics, but when all of the metrics were considered, these models performed better. When the success probability was increased to 0.7, the performance of all models was comparable, though the LR had the highest accuracy and AUC.

4.3.3 High Dimension Results

This section compares the performance of the LR, ANN, SVM, and WkNN on high-dimensional data.

4.3.3.1 Accuracy: High Dimension

The accuracy of models fitted to high-dimension datasets was evaluated, and the results are discussed below.

Table 4.25: Accuracy: unbalanced classes and 150 predictors

Sim ID	Reps	Avg. Prob $(y = 1)$	Accuracy				
			Logit	Probit	ANN	SVM	WkNN
S16	50	$\pm$ 0.3	0.51	0.49	0.54	0.58	0.58
	100		0.50	0.51	0.55	0.59	0.58
	500		0.50	0.50	0.55	0.59	0.58
S17	50	$\pm$ 0.5	0.51	0.50	0.43	0.43	0.45
	100		0.52	0.52	0.44	0.43	0.45
	500		0.51	0.51	0.43	0.43	0.44
S18	50	$\pm$ 0.7	0.52	0.52	0.61	0.60	0.61
	100		0.52	0.52	0.62	0.59	0.60
	500		0.52	0.52	0.61	0.59	0.61

In the high-dimensional case, the LR was outperformed by the SVM, WkNN, and ANN respectively for unbalanced response classes (0.3 probability of success). However, as the probability of success increased to 0.7, the ANN and WkNN outperformed

the LR and SVM with the highest accuracy. LR outperformed the machine learning models when the response classes were balanced, although the LR's accuracy was also very low.

4.3.3.2 Kappa: High Dimension

Using the Kappa statistic, the performance of the models fitted to the high-dimension datasets is discussed below.

Table 4.26: Kappa: unbalanced classes and 150 predictors

Sim ID	Reps	Avg. Prob $(y = 1)$	Kappa statistic				
			Logit	Probit	ANN	SVM	WkNN
S16	50	$\pm$ 0.3	0.01	-0.02	0.05	0.13	0.11
	100		0.00	0.01	0.07	0.15	0.11
	500		0.01	0.00	0.06	0.15	0.11
S17	50	$\pm$ 0.5	0.02	0.01	-0.14	-0.14	-0.12
	100		0.04	0.04	-0.12	-0.14	-0.12
	500		0.03	0.03	-0.14	-0.14	-0.13
S18	50	$\pm$ 0.7	0.03	0.03	0.03	0.02	-0.12
	100		0.02	0.03	0.04	0.02	-0.13
	500		0.03	0.02	0.04	0.01	-0.12

The machine learning models also outperformed the LR with Kappa statistics when the probability of success was set to 0.3, with the SVM outperforming all other models with a Kappa statistic of 0.15 indicating none to slight agreement between predicted and observed classes. The SVM was followed by the WkNN which had a 0.11 Kappa statistic, and the ANN with 0.06. The logit models' Kappa statistic was 0.01 while that of the probit model was 0. This was one of the few instances where the performance of these two models differed slightly. In unbalanced data with a success probability of 0.7, the ANN outperformed all competing models with a Kappa value of 0.04, followed by the logit model with 0.03, the probit model with 0.02, and the SVM with 0.01. The WkNN performed the worst, as measured by the lowest Kappa statistic. However, when the response classes were balanced, the LR model, both the logit and probit

models, outperformed the machine learning models that produced a Kappa statistic less than 0, while the LR model produced a Kappa statistic of 0.03.

4.3.3.3 AUC: High Dimension

Using the AUC, the performance of the models fitted to the high-dimension datasets is discussed below.

Table 4.27: AUC: unbalanced classes and 150 predictors

Sim ID	Reps	Avg. Prob $(y = 1)$	AUC				
			Logit	Probit	ANN	SVM	WkNN
S16	50	$\pm$ 0.3	0.51	0.50	0.54	0.49	0.53
	100		0.50	0.50	0.53	0.48	0.52
	500		0.50	0.50	0.53	0.49	0.52
S17	50	$\pm$ 0.5	0.51	0.53	0.40	0.55	0.42
	100		0.51	0.51	0.40	0.56	0.43
	500		0.52	0.52	0.40	0.56	0.42
S18	50	$\pm$ 0.7	0.53	0.52	0.51	0.43	0.41
	100		0.51	0.52	0.51	0.43	0.41
	500		0.51	0.51	0.51	0.43	0.41

The models performed poorly in separating classes, as evidenced by very low AUC values. Nevertheless, the ANN and WkNN outperformed the LR and SVM for unbalanced datasets when the success probability was 0.3. Once the probability of success was increased to 0.7, both the LR and the ANN performed better with 51% AUC. The SVM (56% AUC) outperformed the competing models for balanced classes, followed by the LR (52% AUC).

4.3.3.4 Sensitivity and Specificity: High Dimension

Using sensitivity and specificity, the performance of the models fitted to the high-dimension datasets is discussed below.

Table 4.28: Sensitivity and specificity: unbalanced classes and 150 predictors

Sim ID	Reps	Pr($y = 1$)	Sensitivity					Specificity				
			Logit	Probit	ANN	SVM	WkNN	Logit	Probit	ANN	SVM	WkNN
S16	50	$\pm$ 0.3	0.52	0.52	0.65	0.90	0.74	0.50	0.49	0.41	0.08	0.37
	100		0.50	0.52	0.66	0.90	0.74	0.48	0.49	0.41	0.09	0.37
	500		0.50	0.51	0.66	0.91	0.74	0.49	0.49	0.41	0.08	0.37
S17	50	$\pm$ 0.5	0.51	0.52	0.38	0.37	0.38	0.51	0.52	0.47	0.71	0.50
	100		0.51	0.51	0.40	0.38	0.38	0.51	0.50	0.47	0.70	0.50
	500		0.52	0.52	0.40	0.38	0.37	0.51	0.51	0.47	0.71	0.50
S18	50	$\pm$ 0.7	0.51	0.50	0.27	0.01	0.05	0.55	0.53	0.75	0.98	0.84
	100		0.49	0.50	0.27	0.00	0.05	0.53	0.54	0.76	0.98	0.85
	500		0.50	0.50	0.28	0.00	0.05	0.53	0.53	0.76	0.98	0.85

On the unbalanced dataset, the machine learning models had the highest sensitivity when the success probability was 0.3 and the highest specificity when the success probability was 0.7. For balanced classes, the SVM had the highest specificity (71%) among competing models, although its sensitivity (38%) was lower than that of LR and ANN. In both balanced and unbalanced datasets, the LR's results were comparable to a random guess.

Summary

The machine learning models performed better on high-dimensional and unbalanced data than the LR, especially the ANN when the success probability was 0.7. However, the LR performed better with balanced data, despite being outperformed by the SVM in terms of the AUC.

4.3.4 Predictor Selection Results

This section presents the results of the LR, ANN, SVM, and WkNN before and after predictor selection. The dataset generated contained ten continuous predictors, $x_1, x_2, ..., x_{10}$. During predictor selection with a balanced dataset, models were trained using 10-fold cross-validation with 50 repeats. The LR with both logit and probit link functions chose only two predictors (x_8 and x_9 respectively) as the only significant predictors, the SVM selected all predictors with x_8, x_9, x_1, x_3, x_5 as the top 5, the NN selected 2 predictors (x_8, x_5), and the WkNN selected only one predictor (x_8). Therefore, the following results compare the models with all 10 predictors before predictor selection to the models with x_8, x_9, and x_5 after predictor selection.

4.3.4.1 Accuracy: Predictor Selection

The accuracy of models fitted before and after predictor selection was evaluated, and the results are discussed below.

Table 4.29: Accuracy: unbalanced classes prior to predictor selection

Sim ID	Reps	Avg. Prob $(y = 1)$	Accuracy				
			Logit	Probit	ANN	SVM	WkNN
S19	50	$\pm$ 0.3	0.81	0.81	0.81	0.81	0.75
	100		0.81	0.81	0.81	0.81	0.75
	500		0.81	0.81	0.81	0.81	0.75
S20	50	$\pm$ 0.5	0.51	0.52	0.51	0.52	0.49
	100		0.51	0.51	0.51	0.52	0.49
	500		0.52	0.51	0.51	0.52	0.49
S21	50	$\pm$ 0.7	0.86	0.86	0.86	0.86	0.84
	100		0.86	0.86	0.86	0.86	0.84
	500		0.86	0.86	0.86	0.86	0.84

LR, ANN, and SVM performed identically in both instances of unbalanced data, outperforming WkNN. When response classes were balanced, the logit and SVM outperformed the other models.

Table 4.30: Accuracy: unbalanced classes after predictor selection

Sim ID	Reps	Avg. Prob $(y = 1)$	Accuracy				
			Logit	Probit	ANN	SVM	WkNN
S22	50	$\pm$ 0.3	0.81	0.81	0.81	0.81	0.79
	100		0.81	0.81	0.81	0.81	0.79
	500		0.81	0.81	0.81	0.81	0.79
S23	50	$\pm$ 0.5	0.53	0.53	0.51	0.52	0.51
	100		0.54	0.54	0.51	0.52	0.51
	500		0.54	0.54	0.51	0.52	0.51
S24	50	$\pm$ 0.7	0.86	0.86	0.86	0.86	0.85
	100		0.86	0.86	0.86	0.86	0.85
	500		0.86	0.86	0.86	0.86	0.85

Predictor selection only improved the LR's accuracy on a balanced dataset; on an unbalanced dataset, the LR's accuracy was unaffected. The WkNN also improved after predictor selection, for both balanced and unbalanced datasets. Predictor selection did not affect the performance of the SVM and the ANN models.

4.3.4.2 Kappa Statistic: Predictor Selection

The Kappa statistics of models fitted before and after predictor selection was evaluated, and the results are discussed below.

Table 4.31: Kappa statistic: unbalanced classes prior to predictor selection

Sim ID	Reps	Avg. Prob $(y = 1)$	Kappa statistic				
			Logit	Probit	ANN	SVM	WkNN
S19	50	$\pm$ 0.3	0.00	0.00	0.02	0.00	0.00
	100		0.00	0.00	0.02	0.00	0.00
	500		0.00	0.00	0.02	0.00	0.00
S20	50	$\pm$ 0.5	0.02	0.03	0.01	0.04	-0.01
	100		0.02	0.02	0.02	0.04	-0.02
	500		0.03	0.02	0.01	0.04	-0.02
S21	50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.03
	100		0.00	0.00	0.00	0.00	0.04
	500		0.00	0.00	0.00	0.00	0.03

The ANN performed better for unbalanced response classes with the highest Kappa statistic when the success probability was 0.3, the SVM performed better for balanced response classes, and the WkNN performed better for unbalanced response classes when a success probability was 0.7.

Table 4.32: Kappa statistic: unbalanced classes after predictor selection

Sim ID	Reps	Avg. Prob $(y = 1)$	Kappa statistic				
			Logit	Probit	ANN	SVM	WkNN
S22	50	$\pm$ 0.3	0.00	0.00	0.00	0.00	0.03
	100		0.00	0.00	0.00	0.00	0.03
	500		0.00	0.00	0.00	0.00	0.03
S23	50	$\pm$ 0.5	0.06	0.06	0.01	0.04	0.02
	100		0.07	0.07	0.01	0.04	0.02
	500		0.07	0.07	0.01	0.04	0.02
S24	50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.02
	100		0.00	0.00	0.00	0.00	0.04
	500		0.00	0.00	0.00	0.00	0.02

After predictor selection for the LR, the Kappa statistic improved for balanced response classes but remained unchanged for unbalanced response classes. The Kappa statistic of the ANN decreased from 0.02 to 0 after predictor selection when response classes were unbalanced (success probability of 0.3). This may be due to the inclusion of x_9 in the model, which the ANN model did not select as one of the significant predictors but was included to ensure the models' comparability. The WkNN's Kappa statistics improved after predictor selection when the success probability was set to 0.3 and 0.5. So, following predictor selection, the LR outperformed machine learning models in a balanced dataset with a Kappa statistic of 0.07, whereas the WkNN outperformed competing models on an unbalanced dataset.

4.3.4.3 AUC: Predictor Selection

The AUC of models fitted before and after predictor selection was evaluated, and the results are discussed below.

Table 4.33: AUC: unbalanced classes prior to predictor selection

Sim ID	Reps	Avg. Prob $(y = 1)$	AUC				
			Logit	Probit	ANN	SVM	WkNN
S19	50	$\pm$ 0.3	0.61	0.61	0.57	0.51	0.52
	100		0.61	0.61	0.56	0.51	0.52
	500		0.61	0.61	0.57	0.51	0.52
S20	50	$\pm$ 0.5	0.52	0.52	0.50	0.49	0.50
	100		0.52	0.52	0.50	0.49	0.50
	500		0.52	0.52	0.50	0.49	0.50
S21	50	$\pm$ 0.7	0.60	0.60	0.53	0.50	0.50
	100		0.60	0.60	0.53	0.50	0.50
	500		0.60	0.60	0.53	0.50	0.50

Based on the preceding AUC results, the LR performed better in both balanced and unbalanced datasets prior to predictor selection.

Table 4.34: AUC: unbalanced classes after predictor selection

Sim ID	Reps	Avg. Prob $(y = 1)$	AUC				
			Logit	Probit	ANN	SVM	WkNN
S22	50	$\pm$ 0.3	0.54	0.54	0.54	0.50	0.56
	100		0.54	0.54	0.54	0.50	0.56
	500		0.54	0.54	0.54	0.50	0.56
S23	50	$\pm$ 0.5	0.54	0.54	0.51	0.52	0.51
	100		0.54	0.54	0.51	0.52	0.51
	500		0.54	0.54	0.51	0.52	0.51
S24	50	$\pm$ 0.7	0.54	0.54	0.52	0.50	0.58
	100		0.54	0.54	0.52	0.50	0.58
	500		0.54	0.54	0.52	0.50	0.58

After predictor selection on a balanced dataset, the AUC of all models increased, whereas it declined for an unbalanced dataset with a success probability of 0.3. For the unbalanced dataset, when the success probability was 0.7, the performance of the LR and ANN declined, while the WkKNN improved and the AUC of the SVM remained unchanged. In general, the LR performed better on the balanced dataset, while the WkNN performed better on the unbalanced dataset.

4.3.4.4 Sensitivity and Specificity: Predictor Selection

The sensitivity and specificity of models fitted before and after predictor selection was evaluated, and the results are discussed below.

Table 4.35: Sensitivity and specificity: unbalanced classes prior to predictor selection

Sim ID	Reps	Pr($y = 1$)	Sensitivity					Specificity				
			Logit	Probit	ANN	SVM	WkNN	Logit	Probit	ANN	SVM	WkNN
S19	50	$\pm$ 0.3	1.00	1.00	0.97	1.00	0.92	0.00	0.00	0.04	0.00	0.08
	100		1.00	1.00	0.97	1.00	0.92	0.00	0.00	0.04	0.00	0.07
	500		1.00	1.00	0.97	1.00	0.92	0.00	0.00	0.04	0.00	0.07
S20	50	$\pm$ 0.5	0.42	0.42	0.48	0.14	0.44	0.60	0.61	0.52	0.84	0.54
	100		0.42	0.42	0.47	0.14	0.44	0.60	0.60	0.53	0.84	0.54
	500		0.42	0.42	0.48	0.14	0.44	0.60	0.61	0.52	0.84	0.54
S21	50	$\pm$ 0.7	0.00	0.00	0.01	0.00	0.05	1.00	1.00	0.99	1.00	0.97
	100		0.00	0.00	0.02	0.00	0.05	1.00	1.00	0.99	1.00	0.97
	500		0.00	0.00	0.02	0.00	0.05	1.00	1.00	0.99	1.00	0.97

Table 4.36: Sensitivity and specificity: unbalanced classes after predictor selection

Sim ID	Reps	Pr($y = 1$)	Sensitivity					Specificity				
			Logit	Probit	ANN	SVM	WkNN	Logit	Probit	ANN	SVM	WkNN
S22	50	$\pm$ 0.3	1.00	1.00	1.00	1.00	0.96	0.00	0.00	0.00	0.00	0.07
	100		1.00	1.00	1.00	1.00	0.96	0.00	0.00	0.00	0.00	0.07
	500		1.00	1.00	1.00	1.00	0.96	0.00	0.00	0.00	0.00	0.07
S23	50	$\pm$ 0.5	0.07	0.07	0.01	0.04	0.02	0.04	0.05	0.05	0.05	0.05
	100		0.07	0.07	0.01	0.04	0.02	0.05	0.04	0.04	0.05	0.04
	500		0.07	0.07	0.01	0.04	0.02	0.05	0.05	0.04	0.05	0.05
S24	50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.03	1.00	1.00	1.00	1.00	0.98
	100		0.00	0.00	0.00	0.00	0.03	1.00	1.00	1.00	1.00	0.98
	500		0.00	0.00	0.00	0.00	0.03	1.00	1.00	1.00	1.00	0.98

After predictor selection on a balanced data set, the sensitivity and specificity of all

models dropped drastically. Neither true success nor true failure was accurately identified by the models. Both the LR and SVM performed identically on the unbalanced datasets, maintaining the same sensitivity and specificity before and after predictor selection. The ANN and WkNN performed similarly on imbalanced datasets, with an improved sensitivity when the probability of success was 0.3 and a decline when the probability was increased to 0.7. On the unbalanced dataset (probability set to 0.3), the specificity of the ANN declined while that of the WkNN remained the same after predictor selection. Their specificity improved on the unbalanced dataset when the probability was set to 0.7.

Summary

Overall, predictor selection improved the performance of the LR on balanced datasets. The performance of this model remained unchanged on unbalanced datasets after predictor selection. The WkNN performed better after predictor selection on the unbalanced datasets, particularly when the probability of success was approximately 0.7. Predictor selection did not affect the performance of the SVM, while that of the ANN either remained the same or worsened.

Chapter 5

Conclusion

5.1 Summary

Different techniques have evolved in response to solving classification problems. These methods include traditional statistical methods and machine learning methods. Though the literature explored the comparisons of these methods, their performance differs depending on the dataset used, leaving a researcher in a difficult position of determining which method will perform best under specific conditions, and this is a challenge that this research attempts to address.

The objectives of this study were to compare the performance of the LR, ANN, SVM, and the WkNN using accuracy, Kappa statistic, AUC, sensitivity, and specificity as performance measures given different data types generated through simulation. The study simulated a total of 24 scenarios, and for each simulation, comparing the models' performance to identify the best-performing model.

The study investigated the effect of binning predictors on model performance and found that binning predictors into a maximum of two classes improved the performance of the LR. The ANN model was better off with numeric predictors. Binning predictors improved the performance of the WkNN model fitted on the balanced dataset when one predictor was binned into two classes or two predictors into three classes.

Once the imbalance was introduced into the dataset, the WkNN model was better off with numeric predictors when the probability of success was 0.3; and when the probability was increased to 0.7, the WkNN model was improved by binning two predictors into two or three classes. The performance of the SVM was unaffected by binning predictors

Despite similar performance across measures on the datasets with skewed predictors, the WkNN outperformed the competing models on balanced and unbalanced (success probability of 0.3) data. The ANN model also together with the WkNN outperformed the LR and SVM when response classes were balanced. All models performed similarly on the unbalanced case where the success probability was set to 0.7, with the LR having the highest accuracy and AUC. The LR appeared to be prone to majority class bias.

The machine learning models performed better on high-dimensional unbalanced data than the LR, particularly the ANN when the probability of success was 0.7. However, with balanced high-dimensional data, the LR performed better, despite being outperformed by the SVM in terms of AUC.

Predictor selection improved the performance of the LR on balanced datasets. The performance of this model remained unchanged on unbalanced datasets after predictor selection. The WkNN performed better after predictor selection on the unbalanced datasets, particularly when the probability of success was approximately 0.7. Predictor selection did not affect the performance of the SVM, while that of the ANN either remained the same or worsened.

5.2 Conclusion

Using various dimension sizes of balanced and unbalanced simulated data, this study compared the performance of the LR to that of the ANN, SVM, and WkNN. It utilized a variety of performance metrics, allowing for merited and verified conclusions.

The study concludes that binning predictors into no more than two classes improves the performance of the LR. Furthermore, the study found that binning predictors improved the performance of the WkNN given that the probability of success was at least 0.5. Binning predictors did not affect the performance of the SVM model while the ANN was better off with numeric predictors.

The results also showed that if predictors are skewed, the WkNN model performed better than the competing models when the probability of success was at most 0.5. Nonetheless, when the probability increases above 0.5, the models perform similarly. In the case of imbalance, the LR and SVM could still produce similar results with fewer predictors, but the performance of the LR fitted on balanced data improved after predictor selection. Once there were more instances of the success class in the data, then the WkNN also improved performance after predictor selection. Predictor selection did not benefit the ANN model.

The study also concludes that machine learning models perform better with high-dimensional unbalanced data than the LR; however, the LR outperforms machine learning models once the data is balanced. The LR with logit and probit links functions perform identically.

5.3 Recommendations

This study suggests that binning predictors be applied to the LR only when the predictors are binned into no more than two classes. If there are more than two classes, the WkNN may be advantageous given that the data is either balanced or has more success occurrences. Binning predictors while using the ANN is not advised. Since binning predictors did not affect the SVM performance, this strategy can be applied in conjunction with the SVM whenever appropriate.

Depending on the researcher's objectives, predictor selection should be applied to the LR and SVM for both balanced and unbalanced data to reduce complexity and improve

interpretability, but this will most likely not improve performance but will retain similar results as the original models. If the data contains a high proportion of the success class then predictor selection is advised for the WkNN as it would likely improve the model performance.

The study further recommends using the WkNN when working with data that contains skewed predictors, particularly when the data is balanced or has fewer instances in the success class. Machine learning models, the ANN, WkNN, and SVM are recommended when working with high-dimensional unbalanced data.

The findings of this study provide insight into selecting the best classification method for a problem based on the objectives and characteristics of the data in question. Applying this approach to compare the performance of other methods such as decision trees and deep neural networks, and using other predictor selection methods such as Lasso would be a promising area of future research. Moreover, altering the structure of the neural network and experimenting with various activation functions to discover the network with the greatest performance could be investigated. In addition, experimenting with different distance functions and kernels for the WkNN could be another aspect to consider. In the case of skewed predictors, the study only looked at the scenario where all predictors were skewed; however, in future, the effect of mixing predictors by skewing some and leaving others normally distributed could be evaluated.

Bibliography

Beck, M. W. (2018). NeuralNetTools: Visualization and analysis tools for neural networks. *Journal of Statistical Software*, 85(11):1–20. 19

Boyacioglu, M. A., Kara, Y., and Baykan, Ö. K. (2009). Predicting bank financial failures using neural networks, support vector machines and multivariate statistical methods: A comparative analysis in the sample of savings deposit insurance fund (sdif) transferred banks in Turkey. *Expert Systems with Applications*, 36(2):3355–3366. 9

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159. 17

Bui, D. T., Tuan, T. A., Klempe, H., Pradhan, B., and Revhaug, I. (2016). Spatial prediction models for shallow landslide hazards: a comparative assessment of the efficacy of support vector machines, artificial neural networks, kernel logistic regression, and logistic model tree. *Landslides*, 13(2):361–378. 10

Byun, H. and Lee, S.-W. (2003). A survey on pattern recognition applications of support vector machines. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(3):459–486. 5

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46. 17

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297. 4

Doran, M., Raicu, D. S., Furst, J. D., Settimi, R., Schipma, M., and Chandler, D. P. (2007). Oligonucleotide microarray identification of bacillus anthracis strains using support vector machines. *Bioinformatics*, 23(4):487–492. 5

Dreiseitl, S. and Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*, 35(5-6):352–359. 4

Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(4):325–327. 4

Eftekhar, B., Mohammad, K., Ardebili, H. E., Ghodsi, M., and Ketabchi, E. (2005). Comparison of artificial neural network and logistic regression models for prediction of mortality in head trauma based on initial clinical data. *BMC Medical Informatics and Decision Making*, 5(1):1–8. 4

El_Jerjawi, N. S. and Abu-Naser, S. S. (2018). Diabetes prediction using artificial neural network. *International Journal of Advanced Science and Technology*, 121:55–64. 10

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422. 22

Hechenbichler, K. and Schliep, K. (2004). Weighted k-nearest-neighbor techniques and ordinal classification. `http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-1769-9`. Online; accessed 30 June 2020. 16

Huang, W., Nakamori, Y., and Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10):2513–2522. 5

Hunt, T. (2020). *ModelMetrics: Rapid Calculation of Model Metrics*. R package version 1.2.2.2. 19

Inayah, N., Lestari, I., and Musti, M. (2020). Comparative analysis of performance k-nearest neighbor and weighted k-nearest neighbor combined with global encoding in predicting protein-protein interaction human immunodeficiency virus type 1 with human. In *AIP Conference Proceedings*, volume 2242, page 030013, Depok, Indonesia. AIP Publishing LLC. 11

Intrator, O. and Intrator, N. (2001). Interpreting neural-network results: a simulation study. *Computational Statistics & Data Analysis*, 37(3):373–393. 5

Izenman, A. J. (2008). Modern multivariate statistical techniques. *Regression, Classification and Manifold Learning*, 10:978–0. 11, 14

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning*, volume 1398, pages 137–142, Berlin, Heidelberg. Springer. 14

Khan, J., Wei, J. S., Ringner, M., Saal, L. H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C. R., Peterson, C., et al. (2001). Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7(6):673–679. 8

Khuman, M. M. B. (2013). Classification of remote sensing data using k-nn method. *J. Inf. Knowl. Res. Electron. Commun. Eng*, 2:817–821. 15

Kim, Y. S. (2008). Comparison of the decision tree, artificial neural network, and linear regression methods based on the number and types of independent variables and sample size. *Expert Systems with Applications*, 34(2):1227–1234. 5

Kim, Y. S. (2010). Performance evaluation for classification methods: A comparative simulation study. *Expert Systems with Applications*, 37(3):2292–2306. 5, 6, 30, 31, 32

Kim, Y. S. (2011). Prediction of hotel bankruptcy using support vector machine, artificial neural network, logistic regression, and multivariate discriminant analysis. *The Service Industries Journal*, 31(3):441–468. 9

Kuhn, M. (2021). *caret: Classification and Regression Training*. R package version 6.0-90. 19

Kumar, U. A. (2005). Comparison of neural networks and regression analysis: A new insight. *Expert Systems with Applications*, 29(2):424–430. 8

Li, S., Wunsch, D. C., O'Hair, E., and Giesselmann, M. G. (2001). Comparative analysis of regression and artificial neural network models for wind turbine power curve estimation. *J. Sol. Energy Eng.*, 123(4):327–332. 5, 13

Lin, Y.-P., Chu, H.-J., Wu, C.-F., and Verburg, P. H. (2011). Predictive ability of logistic regression, auto-logistic regression and neural network models in empirical land-use change modeling–a case study. *International Journal of Geographical Information Science*, 25(1):65–87. 9

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133. 4

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2021). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.7-9. 19

Min, J. H. and Lee, Y.-C. (2005). Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. *Expert Systems with Applications*, 28(4):603–614. 5, 8

Mohsen, H., El-Dahshan, E.-S. A., El-Horbaty, E.-S. M., and Salem, A.-B. M. (2018). Classification using deep learning neural networks for brain tumors. *Future Computing and Informatics Journal*, 3(1):68–71. 10, 13

Morris, T. P., White, I. R., and Crowther, M. J. (2019). Using simulation studies to evaluate statistical methods. *Statistics in Medicine*, 38(11):2074–2102. 20

Muniz, A., Liu, H., Lyons, K., Pahwa, R., Liu, W., Nobre, F., and Nadal, J. (2010). Comparison among probabilistic neural network, support vector machine and logistic regression for evaluating the effect of subthalamic stimulation in Parkinson disease on ground reaction force during gait. *Journal of Biomechanics*, 43(4):720–726. 9

Musa, A. B. (2013). Comparative study on classification performance between support vector machine and logistic regression. *International Journal of Machine Learning and Cybernetics*, 4(1):13–24. 4, 9

Pearson, K. (1904). *On the theory of contingency and its relation to association and normal correlation*. Cambridge University Press, London. 16

Portela González, J., Muñoz San Roque, A., and Pizarroso Gonzalo, J. (2020). *NeuralSens: Sensitivity Analysis of Neural Networks*. R package version 0.2.2. 19

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. 19

Rizwan, M. and Anderson, D. V. (2016). Comparison of distance metrics for phoneme classification based on deep neural network features and weighted k-nn classifier. In *Workshop on Machine Learning in Speech and Language Processing*, San Francisco. 5

Sadri, J., Suen, C. Y., and Bui, T. D. (2003). Application of support vector machines for recognition of handwritten Arabic/Persian digits. In *Proceedings of Second Iranian Conference on Machine Vision and Image Processing*, volume 1, pages 300–307, Tehran. 14

Salas-Eljatib, C., Fuentes-Ramirez, A., Gregoire, T. G., Altamirano, A., and Yaitul, V. (2018). A study on the effects of unbalanced data when fitting logistic regression models in ecology. *Ecological Indicators*, 85:502–508. 10

Schliep, K. and Hechenbichler, K. (2016). *kknn: Weighted k-Nearest Neighbors*. R package version 1.3.1. 19

Stylianou, N., Akbarov, A., Kontopantelis, E., Buchan, I., and Dunn, K. W. (2015). Mortality risk prediction in burn injury: Comparison of logistic regression with machine learning approaches. *Burns*, 41(5):925–934. 10

SubbaNarasimha, P., Arinze, B., and Anandarajan, M. (2000). The predictive accuracy of artificial neural networks and multiple regression in the case of skewed data: Exploration of some issues. *Expert Systems with Applications*, 19(2):117–123. 8

Tay, F. E. and Cao, L. (2001). Application of support vector machines in financial time series forecasting. *Omega*, 29(4):309–317. 14

Tu, J. V. (1996). Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of Clinical Epidemiology*, 49(11):1225–1231. 4

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. ISBN 0-387-95457-0. 19

Wang, Z., Na, J., and Zheng, B. (2020). An improved kNN classifier for epilepsy diagnosis. *IEEE Access*, 8:100022–100030. 11

Wickham, H., Girlich, M., and Ruiz, E. (2021). *dbplyr: A 'dplyr' Back End for Databases*. R package version 2.1.1. 19

Zhao, M. and Chen, J. (2016). Improvement and comparison of weighted k nearest neighbors classifiers for model selection. *Journal of Software Engineering*, 10(1):109–118. 5, 15

Appendix A

A.1 Appendix

Binning Predictors

Median Accuracy

Table A.1: Median accuracy for LR, ANN, SVM, and WkNN models considering unbalanced classes with continuous predictors

Reps	Avg. Prob $(y = 1)$	Accuracy (median)				
		Logit	Probit	ANN	SVM	WkNN
50	$\pm$ 0.3	0.85	0.85	0.85	0.85	0.83
100		0.85	0.85	0.85	0.85	0.83
500		0.85	0.85	0.85	0.85	0.83
50	$\pm$ 0.5	0.51	0.52	0.51	0.53	0.47
100		0.51	0.52	0.51	0.53	0.47
500		0.51	0.51	0.51	0.53	0.47
50	$\pm$ 0.7	0.85	0.85	0.85	0.85	0.83
100		0.85	0.85	0.85	0.85	0.83
500		0.85	0.85	0.85	0.85	0.83

Standard Error of the Accuracy

Table A.2: Standard error of the accuracy for LR, ANN, SVM, and WkNN models considering unbalanced classes with continuous predictors

Reps	Avg. Prob $(y = 1)$	Standard error (accuracy)				
		Logit	Probit	ANN	SVM	WkNN
50	$\pm$ 0.3	0.00	0.00	0.00	0.00	0.00
100		0.00	0.00	0.00	0.00	0.00
500		0.00	0.00	0.00	0.00	0.00
50	$\pm$ 0.5	0.00	0.00	0.00	0.00	0.00
100		0.00	0.00	0.00	0.00	0.00
500		0.00	0.00	0.00	0.00	0.00
50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.00
100		0.00	0.00	0.00	0.00	0.00
500		0.00	0.00	0.00	0.00	0.00

Median Kappa Statistic

Table A.3: Median Kappa statistic for LR, ANN, SVM, and WkNN models considering unbalanced classes with continuous predictors

Reps	Avg. Prob $(y = 1)$	Kappa statistic (median)				
		Logit	Probit	ANN	SVM	WkNN
50	$\pm$ 0.3	0.00	0.00	0.00	0.00	0.04
100		0.00	0.00	0.00	0.00	0.04
500		0.00	0.00	0.00	0.00	0.04
50	$\pm$ 0.5	-0.03	-0.02	-0.01	0.00	-0.07
100		-0.02	-0.02	-0.01	0.00	-0.07
500		-0.02	-0.02	-0.01	0.00	-0.07
50	$\pm$ 0.7	0.00	0.00	0.00	0.00	-0.02
100		0.00	0.00	0.00	0.00	-0.02
500		0.00	0.00	0.00	0.00	-0.02

Standard Error of the Kappa Statistic

Table A.4: Standard error of the Kappa statistic for LR, ANN, SVM, and WkNN models considering unbalanced classes with continuous predictors

Reps	Avg. Prob $(y = 1)$	Standard error (Kappa statistic)				
		Logit	Probit	ANN	SVM	WkNN
50	$\pm$ 0.3	0.00	0.00	0.00	0.00	0.00
100		0.00	0.00	0.00	0.00	0.00
500		0.00	0.00	0.00	0.00	0.00
50	$\pm$ 0.5	0.00	0.00	0.00	0.00	0.00
100		0.00	0.00	0.00	0.00	0.00
500		0.00	0.00	0.00	0.00	0.00
50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.00
100		0.00	0.00	0.00	0.00	0.00
500		0.00	0.00	0.00	0.00	0.00

Median AUC

Table A.5: Median AUC for LR, ANN, SVM, and WkNN models considering unbalanced classes with continuous predictors

Reps	Avg. Prob $(y = 1)$	AUC (median)				
		Logit	Probit	ANN	SVM	WkNN
50	$\pm$ 0.3	0.62	0.62	0.59	0.49	0.56
100		0.62	0.61	0.59	0.50	0.55
500		0.61	0.61	0.59	0.50	0.55
50	$\pm$ 0.5	0.47	0.48	0.49	0.50	0.47
100		0.47	0.47	0.50	0.50	0.47
500		0.48	0.47	0.50	0.49	0.47
50	$\pm$ 0.7	0.51	0.51	0.50	0.50	0.49
100		0.51	0.51	0.50	0.50	0.49
500		0.51	0.51	0.50	0.50	0.49

Standard Error of the AUC

Table A.6: Standard error of the AUC for LR, ANN, SVM, and WkNN models considering unbalanced classes with continuous predictors

Reps	Avg. Prob $(y = 1)$	Standard error (AUC)				
		Logit	Probit	ANN	SVM	WkNN
50	$\pm$ 0.3	0.00	0.00	0.00	0.00	0.00
100		0.00	0.00	0.00	0.00	0.00
500		0.00	0.00	0.00	0.00	0.00
50	$\pm$ 0.5	0.00	0.00	0.00	0.00	0.00
100		0.00	0.00	0.00	0.00	0.00
500		0.00	0.00	0.00	0.00	0.00
50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.00
100		0.00	0.00	0.00	0.00	0.00
500		0.00	0.00	0.00	0.00	0.00

Median Sensitivity and Specificity

Table A.7: Median sensitivity and specificity for LR, ANN, SVM, and WkNN models considering unbalanced classes with continuous predictors

Reps	Pr($y = 1$)	Sensitivity (median)					Specificity (median)				
		Logit	Probit	ANN	SVM	WkNN	Logit	Probit	ANN	SVM	WkNN
50	$\pm$ 0.3	1.00	1.00	1.00	1.00	0.97	0.00	0.00	0.00	0.00	0.06
100		1.00	1.00	1.00	1.00	0.97	0.00	0.00	0.00	0.00	0.06
500		1.00	1.00	1.00	1.00	0.97	0.00	0.00	0.00	0.00	0.06
50	$\pm$ 0.5	0.04	0.04	0.26	0.00	0.43	0.94	0.94	0.73	1.00	0.50
100		0.04	0.03	0.24	0.00	0.43	0.94	0.94	0.75	1.00	0.50
500		0.03	0.03	0.25	0.00	0.43	0.94	0.94	0.74	1.00	0.50
50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.01	1.00	1.00	1.00	1.00	0.98
100		0.00	0.00	0.00	0.00	0.01	1.00	1.00	1.00	1.00	0.98
500		0.00	0.00	0.00	0.00	0.01	1.00	1.00	1.00	1.00	0.98

Sensitivity and Specificity Standard Error

Table A.8: Median sensitivity and specificity for LR, ANN, SVM, and WkNN models considering unbalanced classes with continuous predictors

Reps	Pr($y = 1$)	Standard error (sensitivity)					Standard error (specificity)				
		Logit	Probit	ANN	SVM	WkNN	Logit	Probit	ANN	SVM	WkNN
50	$\pm$ 0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
500		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	$\pm$ 0.5	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.00
100		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
500		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	$\pm$ 0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
500		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Appendix B

Binning Predictors

```
library(caret)
library(kernlab)
library(e1071)
library(nnet)
library(NeuralNetTools)
library(kknn)
library(NeuralSens)
library(ModelMetrics)
library(dbplyr)
library(dplyr)

#Generating data
#sample size
n <- 1000

#coefficients
#For average prob>=0.3
```

```
#b0 <- -0.7
#b1 <- -1
#b2 <- -1
#b3 <- -1

#For average prob>=0.5
#b0 <- 0.001
#b1 <- 0.003
#b2 <- 0.001
#b3 <- 0.02

#For average prob>=0.7
b0 <- 0.7
b1 <- 1
b2 <- 1
b3 <- 1

#randomly generating predictors
x1 <- runif(n, 0, 1)
x2 <- runif(n, 0, 1)
x3 <- runif(n, 0, 1)

#randomly generating an error term
e <- rnorm(n,0,1)
xb <- b0 + b1*x1 + b2*x2 + b3*x3
prob <- 1/(1 + exp(-(xb+e)))
#summary(prob)
```

```
y <- rbinom(n, size = 1, prob = prob)

#data
data <- data.frame(x1,x2,x3,y)

#Make y a factor
data$y <-factor(data[,'y'], labels = c("failure", "
   success"))
write.csv(data, file = "Dataset.two.binned.3c.0.7.csv")

#Binning predictors
#Two classes
data$x1 <- as.factor(ifelse(data$x1 <= 0.5, "A", "B"))
data$x2 <- as.factor(ifelse(data$x2 <= 0.5, "A", "B"))

#Three classes
data$x1 <- as.factor(ifelse(data$x1<=0.25, "A", ifelse(
   data$x1>0.25 & data$x1<=0.5, "B", "C")))
data$x2 <- as.factor(ifelse(data$x2<=0.25, "A", ifelse(
   data$x2>0.25 & data$x2<=0.5, "B", "C")))

#Data summary and distribution
summary(data)
par(mfrow=c(2,2))
hist(data$x1, main="", xlab="x1", col="lightsteelblue3"
   )
```

```
hist(data$x2, main="", xlab="x2", col="lightsteelblue3"
   )
hist(data$x3, main="", xlab="x3", col="lightsteelblue3"
   )
barplot(table(data$x1), main="", xlab="x1", col=c("
   navyblue","gold", "red"))
barplot(table(data$x2), main="", xlab="x2", col=c("
   navyblue","gold", "red"))
barplot(table(data$y), xlab="y",col=c("navyblue","gold"
   ) )

#Cross validation
#K-fold CV
data_cv10 <- trainControl(method = "cv", number = 10)

#data_cv10 <- trainControl(method = "cv", number = 10,
   summaryFunction = twoClassSummary, classProbs = TRUE
   ,
#verboseIter = TRUE)

#Logistic regression (logit)
repeats=500 #takes on the values 50, 100, 500

results_LR.logit <- matrix(NA, nrow=repeats, ncol=2)

for (i in 1:repeats) {
```

```
Model_LR.logit <- train(y ~. ,                    # model
   to fit
data = data,
trControl = data_cv10,      # folds
method = "glm",
family="binomial",          # specifying regression
   model
na.action = na.pass)

#Model results
results_LR.logit[i,1] <- Model_LR.logit$results[1,2]
results_LR.logit[i,2] <- Model_LR.logit$results[1,3]
}
confusionMatrix(Model_LR.logit)
write.csv(results_LR.logit,file="results.LR.logit.
   unbinned.500.0.3.csv")

#Logistic regression (probit)
results_LR.probit <- matrix(NA, nrow=repeats, ncol=2)
for (i in 1:repeats) {
Model_LR.probit <- train(y ~. ,
                                    # model to fit
data = data,
trControl = data_cv10,                  # folds
method = "glm",
family="binomial"(link = "probit"),    # specifying
   regression model
```

```
na.action = na.pass)
#Model results
results_LR.probit[i,1] <- Model_LR.probit$results[1,2]
results_LR.probit[i,2] <- Model_LR.probit$results[1,3]
}
write.csv(results_LR.probit,file="results.probit.
   unbinned.500.0.3.csv")

#Neural net
results_nn <- matrix(NA, nrow=repeats, ncol=2)
for (i in 1:repeats) {

Model_nn <- train(y ~. ,                                    #
    model to fit
data = data,
trControl = data_cv10,        # folds
method = "nnet",              # specifying model
na.action = na.pass)
#Model results
results_nn[i,1] <- Model_nn$results[5,3]
results_nn[i,2] <- Model_nn$results[5,4]
}
write.csv(results_nn,file="results.nn.unbinned.500.0.3.
   csv")
#size 3, decay=0.0001, gave the best accuracy and kappa

#SVM
```

```
results_svm <- matrix(NA, nrow=repeats, ncol=2)

for (i in 1:repeats) {

Model_svm <- train(y ~. ,                                    #
   model to fit
data = data,
trControl = data_cv10,                  # folds
method = "svmLinear",                   # specifying model
na.action = na.pass)
#Model results
results_svm[i,1] <- Model_svm$results[1,2]
results_svm[i,2] <- Model_svm$results[1,3]
}
write.csv(results_svm, file = "results_svm.unbinned
   .500.0.3.csv")

#WKNN
results_wknn <- matrix(NA, nrow=repeats, ncol=2)

for (i in 1:repeats) {

Model_wknn <- train(y ~. ,                                   #
   model to fit
data = data,
trControl = data_cv10,                  # folds
method = "kknn",                        # specifying model
```

```
na.action = na.pass)
#Model results
results_wknn[i,1] <- Model_wknn$results[3,4]
results_wknn[i,2] <- Model_wknn$results[3,5]
}
Model_wknn$results
write.csv(results_wknn, file = "results_wknn.unbinned
   .500.0.3.csv")
#Kernel-optimal, k-9, distance-2

#Results
Data.50.reps.Acc <- data.frame(results_LR.logit[,1],
   results_LR.probit[,1], results_nn[,1], results_svm
   [,1], results_wknn[,1])
Data.50.reps.K <- data.frame(results_LR.logit[,2],
   results_LR.probit[,2], results_nn[,2], results_svm
   [,2], results_wknn[,2])

#Accuracy
names(Data.50.reps.Acc)[names(Data.50.reps.Acc) == "
   results_LR.logit...1."] <- "Logit"
names(Data.50.reps.Acc)[names(Data.50.reps.Acc) == "
   results_LR.probit...1."] <- "Probit"
names(Data.50.reps.Acc)[names(Data.50.reps.Acc) == "
   results_nn...1."] <- "NN"
names(Data.50.reps.Acc)[names(Data.50.reps.Acc) == "
   results_svm...1."] <- "SVM"
```

```
names(Data.50.reps.Acc)[names(Data.50.reps.Acc) == "
   results_wknn...1."] <- "WKNN"

Mean.50.reps.acc <- c(mean(Data.50.reps.Acc$Logit),
   mean(Data.50.reps.Acc$Probit), mean(Data.50.reps.Acc
   $NN), mean(Data.50.reps.Acc$SVM), mean(Data.50.reps.
   Acc$WKNN))
Median.50.reps.acc <- c(median(Data.50.reps.Acc$Logit),
    median(Data.50.reps.Acc$Probit), median(Data.50.
   reps.Acc$NN), median(Data.50.reps.Acc$SVM), median(
   Data.50.reps.Acc$WKNN))
Std.error.50.reps.acc <- c(std.error(Data.50.reps.Acc$
   Logit), std.error(Data.50.reps.Acc$Probit), std.
   error(Data.50.reps.Acc$NN), std.error(Data.50.reps.
   Acc$SVM), std.error(Data.50.reps.Acc$WKNN))

#Kappa
names(Data.50.reps.K)[names(Data.50.reps.K) == "results
   _LR.logit...2."] <- "Logit"
names(Data.50.reps.K)[names(Data.50.reps.K) == "results
   _LR.probit...2."] <- "Probit"
names(Data.50.reps.K)[names(Data.50.reps.K) == "results
   _nn...2."] <- "NN"
names(Data.50.reps.K)[names(Data.50.reps.K) == "results
   _svm...2."] <- "SVM"
names(Data.50.reps.K)[names(Data.50.reps.K) == "results
   _wknn...2."] <- "WKNN"
```

```
Mean.50.reps.k <- c(mean(Data.50.reps.K$Logit), mean(
   Data.50.reps.K$Probit), mean(Data.50.reps.K$NN),
   mean(Data.50.reps.K$SVM), mean(Data.50.reps.K$WKNN))
Median.50.reps.k <- c(median(Data.50.reps.K$Logit),
   median(Data.50.reps.K$Probit), median(Data.50.reps.K
   $NN), median(Data.50.reps.K$SVM), median(Data.50.
   reps.K$WKNN))
Std.error.50.reps.k <- c(std.error(Data.50.reps.K$Logit
   ), std.error(Data.50.reps.K$Probit), std.error(Data
   .50.reps.K$NN), std.error(Data.50.reps.K$SVM), std.
   error(Data.50.reps.K$WKNN))
results.reps.50 <- format(round(data.frame(Mean.50.reps
   .acc, Median.50.reps.acc, Std.error.50.reps.acc,
Mean.50.reps.k, Median.50.reps.k, Std.error.50.reps.k)
   ,2),nsmall=2)
write.csv(results.reps.50, file = "results.reps.50.0.3.
   p.csv")

#REPEATS=100
#Results
Data.100.reps.Acc <- data.frame(results_LR.logit[,1],
   results_LR.probit[,1], results_nn[,1], results_svm
   [,1], results_wknn[,1])
Data.100.reps.K <- data.frame(results_LR.logit[,2],
   results_LR.probit[,2], results_nn[,2], results_svm
   [,2], results_wknn[,2])
```

```
#Accuracy
names(Data.100.reps.Acc)[names(Data.100.reps.Acc) == "
   results_LR.logit...1."] <- "Logit"
names(Data.100.reps.Acc)[names(Data.100.reps.Acc) == "
   results_LR.probit...1."] <- "Probit"
names(Data.100.reps.Acc)[names(Data.100.reps.Acc) == "
   results_nn...1."] <- "NN"
names(Data.100.reps.Acc)[names(Data.100.reps.Acc) == "
   results_svm...1."] <- "SVM"
names(Data.100.reps.Acc)[names(Data.100.reps.Acc) == "
   results_wknn...1."] <- "WKNN"

Mean.100.reps.acc <- c(mean(Data.100.reps.Acc$Logit),
   mean(Data.100.reps.Acc$Probit), mean(Data.100.reps.
   Acc$NN), mean(Data.100.reps.Acc$SVM), mean(Data.100.
   reps.Acc$WKNN))
Median.100.reps.acc <- c(median(Data.100.reps.Acc$Logit
   ), median(Data.100.reps.Acc$Probit), median(Data
   .100.reps.Acc$NN), median(Data.100.reps.Acc$SVM),
   median(Data.100.reps.Acc$WKNN))
Std.error.100.reps.acc <- c(std.error(Data.100.reps.Acc
   $Logit), std.error(Data.100.reps.Acc$Probit), std.
   error(Data.100.reps.Acc$NN), std.error(Data.100.reps
   .Acc$SVM), std.error(Data.100.reps.Acc$WKNN))

#Kappa
```

```
names(Data.100.reps.K)[names(Data.100.reps.K) == "
   results_LR.logit...2."] <- "Logit"
names(Data.100.reps.K)[names(Data.100.reps.K) == "
   results_LR.probit...2."] <- "Probit"
names(Data.100.reps.K)[names(Data.100.reps.K) == "
   results_nn...2."] <- "NN"
names(Data.100.reps.K)[names(Data.100.reps.K) == "
   results_svm...2."] <- "SVM"
names(Data.100.reps.K)[names(Data.100.reps.K) == "
   results_wknn...2."] <- "WKNN"

Mean.100.reps.k <- c(mean(Data.100.reps.K$Logit), mean(
   Data.100.reps.K$Probit), mean(Data.100.reps.K$NN),
   mean(Data.100.reps.K$SVM), mean(Data.100.reps.K$WKNN
   ))
Median.100.reps.k <- c(median(Data.100.reps.K$Logit),
   median(Data.100.reps.K$Probit), median(Data.100.reps
   .K$NN), median(Data.100.reps.K$SVM), median(Data
   .100.reps.K$WKNN))
Std.error.100.reps.k <- c(std.error(Data.100.reps.K$
   Logit), std.error(Data.100.reps.K$Probit), std.error
   (Data.100.reps.K$NN), std.error(Data.100.reps.K$SVM)
   , std.error(Data.100.reps.K$WKNN))
results.reps.100 <- format(round(data.frame(Mean.100.
   reps.acc, Median.100.reps.acc, Std.error.100.reps.
   acc,
```

```
Mean.100.reps.k, Median.100.reps.k, Std.error.100.reps.
   k),2),nsmall=2)
write.csv(results.reps.100, file = "results.reps
   .100.0.3.p.csv")

#REPEATS=500
#Results
Data.500.reps.Acc <- data.frame(results_LR.logit[,1],
   results_LR.probit[,1], results_nn[,1], results_svm
   [,1], results_wknn[,1])
Data.500.reps.K <- data.frame(results_LR.logit[,2],
   results_LR.probit[,2], results_nn[,2], results_svm
   [,2], results_wknn[,2])

#Accuracy
names(Data.500.reps.Acc)[names(Data.500.reps.Acc) == "
   results_LR.logit...1."] <- "Logit"
names(Data.500.reps.Acc)[names(Data.500.reps.Acc) == "
   results_LR.probit...1."] <- "Probit"
names(Data.500.reps.Acc)[names(Data.500.reps.Acc) == "
   results_nn...1."] <- "NN"
names(Data.500.reps.Acc)[names(Data.500.reps.Acc) == "
   results_svm...1."] <- "SVM"
names(Data.500.reps.Acc)[names(Data.500.reps.Acc) == "
   results_wknn...1."] <- "WKNN"
```

```
Mean.500.reps.acc <- c(mean(Data.500.reps.Acc$Logit),
   mean(Data.500.reps.Acc$Probit), mean(Data.500.reps.
   Acc$NN), mean(Data.500.reps.Acc$SVM), mean(Data.500.
   reps.Acc$WKNN))
Median.500.reps.acc <- c(median(Data.500.reps.Acc$Logit
   ), median(Data.500.reps.Acc$Probit), median(Data
   .500.reps.Acc$NN), median(Data.500.reps.Acc$SVM),
   median(Data.500.reps.Acc$WKNN))
Std.error.500.reps.acc <- c(std.error(Data.500.reps.Acc
   $Logit), std.error(Data.500.reps.Acc$Probit), std.
   error(Data.500.reps.Acc$NN), std.error(Data.500.reps
   .Acc$SVM), std.error(Data.500.reps.Acc$WKNN))

#Kappa
names(Data.500.reps.K)[names(Data.500.reps.K) == "
   results_LR.logit...2."] <- "Logit"
names(Data.500.reps.K)[names(Data.500.reps.K) == "
   results_LR.probit...2."] <- "Probit"
names(Data.500.reps.K)[names(Data.500.reps.K) == "
   results_nn...2."] <- "NN"
names(Data.500.reps.K)[names(Data.500.reps.K) == "
   results_svm...2."] <- "SVM"
names(Data.500.reps.K)[names(Data.500.reps.K) == "
   results_wknn...2."] <- "WKNN"

Mean.500.reps.k <- c(mean(Data.500.reps.K$Logit), mean(
   Data.500.reps.K$Probit), mean(Data.500.reps.K$NN),
```

```
   mean(Data.500.reps.K$SVM), mean(Data.500.reps.K$WKNN
   ))
Median.500.reps.k <- c(median(Data.500.reps.K$Logit),
   median(Data.500.reps.K$Probit), median(Data.500.reps
   .K$NN), median(Data.500.reps.K$SVM), median(Data
   .500.reps.K$WKNN))
Std.error.500.reps.k <- c(std.error(Data.500.reps.K$
   Logit), std.error(Data.500.reps.K$Probit), std.error
   (Data.500.reps.K$NN), std.error(Data.500.reps.K$SVM)
   , std.error(Data.500.reps.K$WKNN))

results.reps.500 <- format(round(data.frame(Mean.500.
   reps.acc, Median.500.reps.acc, Std.error.500.reps.
   acc,
Mean.500.reps.k, Median.500.reps.k, Std.error.500.reps.
   k),2),nsmall=2)

write.csv(results.reps.500, file = "results.reps
   .500.0.3.p.csv")

#Two summary results
#K-fold CV

data_cv10 <- trainControl(method = "cv", number = 10,
   summaryFunction = twoClassSummary, classProbs = TRUE
   ,
verboseIter = TRUE)
```

```
#Logistic regression (logit)

repeats=500 #takes on the values 50, 100, 500

results_LR.logit <- matrix(NA, nrow=repeats, ncol=3)

for (i in 1:repeats) {

Model_LR.logit <- train(y ~. ,                    # model
   to fit
data = data,
trControl = data_cv10,      # folds
method = "glm",
family="binomial",          # specifying regression
   model
na.action = na.pass)

#Model results
results_LR.logit[i,1] <- Model_LR.logit$results[1,2]
results_LR.logit[i,2] <- Model_LR.logit$results[1,3]
results_LR.logit[i,3] <- Model_LR.logit$results[1,4]
}
Model_LR.logit$results
write.csv(results_LR.logit,file="results.LR.logit.
   unbinned.500.0.3.csv")
```

```
#Logistic regression (probit)
results_LR.probit <- matrix(NA, nrow=repeats, ncol=3)

for (i in 1:repeats) {
Model_LR.probit <- train(y ~. ,
                                    # model to fit
data = data,
trControl = data_cv10,                  # folds
method = "glm",
family="binomial"(link = "probit"),    # specifying
   regression model
na.action = na.pass)
#Model results
results_LR.probit[i,1] <- Model_LR.probit$results[1,2]
results_LR.probit[i,2] <- Model_LR.probit$results[1,3]
results_LR.probit[i,3] <- Model_LR.probit$results[1,4]
}
write.csv(results_LR.probit,file="results.probit.
   unbinned.500.0.3.csv")

#Neural net
results_nn <- matrix(NA, nrow=repeats, ncol=3)

for (i in 1:repeats) {

Model_nn <- train(y ~. ,                                    #
    model to fit
```

```
data = data,
trControl = data_cv10,        # folds
method = "nnet",              # specifying model
na.action = na.pass)
#Model results
results_nn[i,1] <- Model_nn$results[5,3]
results_nn[i,2] <- Model_nn$results[5,4]
results_nn[i,3] <- Model_nn$results[5,5]
}
write.csv(results_nn,file="results.nn.unbinned.500.0.3.
   csv")
#size 3&5, decay=0.1, gave the best ROC and Spec (0.3p)
    but we keep size 3 with decay=0.0001

#SVM
results_svm <- matrix(NA, nrow=repeats, ncol=3)
for (i in 1:repeats) {

Model_svm <- train(y ~. ,                              #
   model to fit
data = data,
trControl = data_cv10,             # folds
method = "svmLinear",              # specifying model
na.action = na.pass)
#Model results
results_svm[i,1] <- Model_svm$results[1,2]
results_svm[i,2] <- Model_svm$results[1,3]
```

```
results_svm[i,3] <- Model_svm$results[1,4]
}
write.csv(results_svm, file = "results_svm.unbinned
   .500.0.3.csv")
#C=1, P=0.3

#WKNN
results_wknn <- matrix(NA, nrow=repeats, ncol=3)

for (i in 1:repeats) {

Model_wknn <- train(y ~. ,                              #
   model to fit
data = data,
trControl = data_cv10,              # folds
method = "kknn",                    # specifying model
na.action = na.pass)
#Model results
results_wknn[i,1] <- Model_wknn$results[3,4]
results_wknn[i,2] <- Model_wknn$results[3,5]
results_wknn[i,3] <- Model_wknn$results[3,6]
}
#Model_wknn$results
write.csv(results_wknn, file = "results_wknn.unbinned
   .500.0.3.csv")
#Kernel-optimal, k-9, distane-2 (0.3p)
```

```
#Results
Data.50.reps.ROC <- data.frame(results_LR.logit[,1],
   results_LR.probit[,1], results_nn[,1], results_svm
   [,1], results_wknn[,1])
Data.50.reps.SENS <- data.frame(results_LR.logit[,2],
   results_LR.probit[,2], results_nn[,2], results_svm
   [,2], results_wknn[,2])
Data.50.reps.SPEC <- data.frame(results_LR.logit[,3],
   results_LR.probit[,3], results_nn[,3], results_svm
   [,3], results_wknn[,3])

#ROC
names(Data.50.reps.ROC)[names(Data.50.reps.ROC) == "
   results_LR.logit...1."] <- "Logit"
names(Data.50.reps.ROC)[names(Data.50.reps.ROC) == "
   results_LR.probit...1."] <- "Probit"
names(Data.50.reps.ROC)[names(Data.50.reps.ROC) == "
   results_nn...1."] <- "NN"
names(Data.50.reps.ROC)[names(Data.50.reps.ROC) == "
   results_svm...1."] <- "SVM"
names(Data.50.reps.ROC)[names(Data.50.reps.ROC) == "
   results_wknn...1."] <- "WKNN"

Mean.50.reps.ROC <- c(mean(Data.50.reps.ROC$Logit),
   mean(Data.50.reps.ROC$Probit), mean(Data.50.reps.ROC
   $NN), mean(Data.50.reps.ROC$SVM), mean(Data.50.reps.
   ROC$WKNN))
```

```
Median.50.reps.ROC <- c(median(Data.50.reps.ROC$Logit),
    median(Data.50.reps.ROC$Probit), median(Data.50.
   reps.ROC$NN), median(Data.50.reps.ROC$SVM), median(
   Data.50.reps.ROC$WKNN))
Std.error.50.reps.ROC <- c(std.error(Data.50.reps.ROC$
   Logit), std.error(Data.50.reps.ROC$Probit), std.
   error(Data.50.reps.ROC$NN), std.error(Data.50.reps.
   ROC$SVM), std.error(Data.50.reps.ROC$WKNN))

#SENSITIVITY
names(Data.50.reps.SENS)[names(Data.50.reps.SENS) == "
   results_LR.logit...2."] <- "Logit"
names(Data.50.reps.SENS)[names(Data.50.reps.SENS) == "
   results_LR.probit...2."] <- "Probit"
names(Data.50.reps.SENS)[names(Data.50.reps.SENS) == "
   results_nn...2."] <- "NN"
names(Data.50.reps.SENS)[names(Data.50.reps.SENS) == "
   results_svm...2."] <- "SVM"
names(Data.50.reps.SENS)[names(Data.50.reps.SENS) == "
   results_wknn...2."] <- "WKNN"

Mean.50.reps.SENS <- c(mean(Data.50.reps.SENS$Logit),
   mean(Data.50.reps.SENS$Probit), mean(Data.50.reps.
   SENS$NN), mean(Data.50.reps.SENS$SVM), mean(Data.50.
   reps.SENS$WKNN))
Median.50.reps.SENS <- c(median(Data.50.reps.SENS$Logit
   ), median(Data.50.reps.SENS$Probit), median(Data.50.
```

```
   reps.SENS$NN), median(Data.50.reps.SENS$SVM), median
   (Data.50.reps.SENS$WKNN))
Std.error.50.reps.SENS <- c(std.error(Data.50.reps.SENS
   $Logit), std.error(Data.50.reps.SENS$Probit), std.
   error(Data.50.reps.SENS$NN), std.error(Data.50.reps.
   SENS$SVM), std.error(Data.50.reps.SENS$WKNN))

#SPECIFICITY
names(Data.50.reps.SPEC)[names(Data.50.reps.SPEC) == "
   results_LR.logit...3."] <- "Logit"
names(Data.50.reps.SPEC)[names(Data.50.reps.SPEC) == "
   results_LR.probit...3."] <- "Probit"
names(Data.50.reps.SPEC)[names(Data.50.reps.SPEC) == "
   results_nn...3."] <- "NN"
names(Data.50.reps.SPEC)[names(Data.50.reps.SPEC) == "
   results_svm...3."] <- "SVM"
names(Data.50.reps.SPEC)[names(Data.50.reps.SPEC) == "
   results_wknn...3."] <- "WKNN"

Mean.50.reps.SPEC <- c(mean(Data.50.reps.SPEC$Logit),
   mean(Data.50.reps.SPEC$Probit), mean(Data.50.reps.
   SPEC$NN), mean(Data.50.reps.SPEC$SVM), mean(Data.50.
   reps.SPEC$WKNN))
Median.50.reps.SPEC <- c(median(Data.50.reps.SPEC$Logit
   ), median(Data.50.reps.SPEC$Probit), median(Data.50.
   reps.SPEC$NN), median(Data.50.reps.SPEC$SVM), median
   (Data.50.reps.SPEC$WKNN))
```

```
Std.error.50.reps.SPEC <- c(std.error(Data.50.reps.SPEC
   $Logit), std.error(Data.50.reps.SPEC$Probit), std.
   error(Data.50.reps.SPEC$NN), std.error(Data.50.reps.
   SPEC$SVM), std.error(Data.50.reps.SPEC$WKNN))

results.reps.50 <- format(round(data.frame(Mean.50.reps
   .ROC, Median.50.reps.ROC, Std.error.50.reps.ROC,
Mean.50.reps.SENS, Median.50.reps.SENS, Std.error.50.
   reps.SENS,
Mean.50.reps.SPEC, Median.50.reps.SPEC, Std.error.50.
   reps.SPEC ),2),nsmall=2)
write.csv(results.reps.50, file = "results.reps.50.0.3.
   p.csv")

#REPEATS=100

#Results
Data.100.reps.ROC <- data.frame(results_LR.logit[,1],
   results_LR.probit[,1], results_nn[,1], results_svm
   [,1], results_wknn[,1])
Data.100.reps.SENS <- data.frame(results_LR.logit[,2],
   results_LR.probit[,2], results_nn[,2], results_svm
   [,2], results_wknn[,2])
Data.100.reps.SPEC <- data.frame(results_LR.logit[,3],
   results_LR.probit[,3], results_nn[,3], results_svm
   [,3], results_wknn[,3])
```

```
#ROC
names(Data.100.reps.ROC)[names(Data.100.reps.ROC) == "
   results_LR.logit...1."] <- "Logit"
names(Data.100.reps.ROC)[names(Data.100.reps.ROC) == "
   results_LR.probit...1."] <- "Probit"
names(Data.100.reps.ROC)[names(Data.100.reps.ROC) == "
   results_nn...1."] <- "NN"
names(Data.100.reps.ROC)[names(Data.100.reps.ROC) == "
   results_svm...1."] <- "SVM"
names(Data.100.reps.ROC)[names(Data.100.reps.ROC) == "
   results_wknn...1."] <- "WKNN"

Mean.100.reps.ROC <- c(mean(Data.100.reps.ROC$Logit),
   mean(Data.100.reps.ROC$Probit), mean(Data.100.reps.
   ROC$NN), mean(Data.100.reps.ROC$SVM), mean(Data.100.
   reps.ROC$WKNN))
Median.100.reps.ROC <- c(median(Data.100.reps.ROC$Logit
   ), median(Data.100.reps.ROC$Probit), median(Data
   .100.reps.ROC$NN), median(Data.100.reps.ROC$SVM),
   median(Data.100.reps.ROC$WKNN))
Std.error.100.reps.ROC <- c(std.error(Data.100.reps.ROC
   $Logit), std.error(Data.100.reps.ROC$Probit), std.
   error(Data.100.reps.ROC$NN), std.error(Data.100.reps
   .ROC$SVM), std.error(Data.100.reps.ROC$WKNN))

#SENSITIVITY
```

```
names(Data.100.reps.SENS)[names(Data.100.reps.SENS) ==
   "results_LR.logit...2."] <- "Logit"
names(Data.100.reps.SENS)[names(Data.100.reps.SENS) ==
   "results_LR.probit...2."] <- "Probit"
names(Data.100.reps.SENS)[names(Data.100.reps.SENS) ==
   "results_nn...2."] <- "NN"
names(Data.100.reps.SENS)[names(Data.100.reps.SENS) ==
   "results_svm...2."] <- "SVM"
names(Data.100.reps.SENS)[names(Data.100.reps.SENS) ==
   "results_wknn...2."] <- "WKNN"

Mean.100.reps.SENS <- c(mean(Data.100.reps.SENS$Logit),
    mean(Data.100.reps.SENS$Probit), mean(Data.100.reps
   .SENS$NN), mean(Data.100.reps.SENS$SVM), mean(Data
   .100.reps.SENS$WKNN))
Median.100.reps.SENS <- c(median(Data.100.reps.SENS$
   Logit), median(Data.100.reps.SENS$Probit), median(
   Data.100.reps.SENS$NN), median(Data.100.reps.SENS$
   SVM), median(Data.100.reps.SENS$WKNN))
Std.error.100.reps.SENS <- c(std.error(Data.100.reps.
   SENS$Logit), std.error(Data.100.reps.SENS$Probit),
   std.error(Data.100.reps.SENS$NN), std.error(Data
   .100.reps.SENS$SVM), std.error(Data.100.reps.SENS$
   WKNN))

#SPECIFICITY
```

```
names(Data.100.reps.SPEC)[names(Data.100.reps.SPEC) ==
   "results_LR.logit...3."] <- "Logit"
names(Data.100.reps.SPEC)[names(Data.100.reps.SPEC) ==
   "results_LR.probit...3."] <- "Probit"
names(Data.100.reps.SPEC)[names(Data.100.reps.SPEC) ==
   "results_nn...3."] <- "NN"
names(Data.100.reps.SPEC)[names(Data.100.reps.SPEC) ==
   "results_svm...3."] <- "SVM"
names(Data.100.reps.SPEC)[names(Data.100.reps.SPEC) ==
   "results_wknn...3."] <- "WKNN"

Mean.100.reps.SPEC <- c(mean(Data.100.reps.SPEC$Logit),
    mean(Data.100.reps.SPEC$Probit), mean(Data.100.reps
   .SPEC$NN), mean(Data.100.reps.SPEC$SVM), mean(Data
   .100.reps.SPEC$WKNN))
Median.100.reps.SPEC <- c(median(Data.100.reps.SPEC$
   Logit), median(Data.100.reps.SPEC$Probit), median(
   Data.100.reps.SPEC$NN), median(Data.100.reps.SPEC$
   SVM), median(Data.100.reps.SPEC$WKNN))
Std.error.100.reps.SPEC <- c(std.error(Data.100.reps.
   SPEC$Logit), std.error(Data.100.reps.SPEC$Probit),
   std.error(Data.100.reps.SPEC$NN), std.error(Data
   .100.reps.SPEC$SVM), std.error(Data.100.reps.SPEC$
   WKNN))

results.reps.100 <- format(round(data.frame(Mean.100.
   reps.ROC, Median.100.reps.ROC, Std.error.100.reps.
```

```
   ROC,
Mean.100.reps.SENS, Median.100.reps.SENS, Std.error
   .100.reps.SENS,
Mean.100.reps.SPEC, Median.100.reps.SPEC, Std.error
   .100.reps.SPEC ),2),nsmall=2)
write.csv(results.reps.100, file = "results.reps
   .100.0.3.p.csv")

#REPEATS=500

#Results
Data.500.reps.ROC <- data.frame(results_LR.logit[,1],
   results_LR.probit[,1], results_nn[,1], results_svm
   [,1], results_wknn[,1])
Data.500.reps.SENS <- data.frame(results_LR.logit[,2],
   results_LR.probit[,2], results_nn[,2], results_svm
   [,2], results_wknn[,2])
Data.500.reps.SPEC <- data.frame(results_LR.logit[,3],
   results_LR.probit[,3], results_nn[,3], results_svm
   [,3], results_wknn[,3])

#ROC
names(Data.500.reps.ROC)[names(Data.500.reps.ROC) == "
   results_LR.logit...1."] <- "Logit"
names(Data.500.reps.ROC)[names(Data.500.reps.ROC) == "
   results_LR.probit...1."] <- "Probit"
```

```
names(Data.500.reps.ROC)[names(Data.500.reps.ROC) == "
   results_nn...1."] <- "NN"
names(Data.500.reps.ROC)[names(Data.500.reps.ROC) == "
   results_svm...1."] <- "SVM"
names(Data.500.reps.ROC)[names(Data.500.reps.ROC) == "
   results_wknn...1."] <- "WKNN"

Mean.500.reps.ROC <- c(mean(Data.500.reps.ROC$Logit),
   mean(Data.500.reps.ROC$Probit), mean(Data.500.reps.
   ROC$NN), mean(Data.500.reps.ROC$SVM), mean(Data.500.
   reps.ROC$WKNN))
Median.500.reps.ROC <- c(median(Data.500.reps.ROC$Logit
   ), median(Data.500.reps.ROC$Probit), median(Data
   .500.reps.ROC$NN), median(Data.500.reps.ROC$SVM),
   median(Data.500.reps.ROC$WKNN))
Std.error.500.reps.ROC <- c(std.error(Data.500.reps.ROC
   $Logit), std.error(Data.500.reps.ROC$Probit), std.
   error(Data.500.reps.ROC$NN), std.error(Data.500.reps
   .ROC$SVM), std.error(Data.500.reps.ROC$WKNN))

#SENSITIVITY
names(Data.500.reps.SENS)[names(Data.500.reps.SENS) ==
   "results_LR.logit...2."] <- "Logit"
names(Data.500.reps.SENS)[names(Data.500.reps.SENS) ==
   "results_LR.probit...2."] <- "Probit"
names(Data.500.reps.SENS)[names(Data.500.reps.SENS) ==
   "results_nn...2."] <- "NN"
```

```
names(Data.500.reps.SENS)[names(Data.500.reps.SENS) ==
   "results_svm...2."] <- "SVM"
names(Data.500.reps.SENS)[names(Data.500.reps.SENS) ==
   "results_wknn...2."] <- "WKNN"

Mean.500.reps.SENS <- c(mean(Data.500.reps.SENS$Logit),
    mean(Data.500.reps.SENS$Probit), mean(Data.500.reps
   .SENS$NN), mean(Data.500.reps.SENS$SVM), mean(Data
   .500.reps.SENS$WKNN))
Median.500.reps.SENS <- c(median(Data.500.reps.SENS$
   Logit), median(Data.500.reps.SENS$Probit), median(
   Data.500.reps.SENS$NN), median(Data.500.reps.SENS$
   SVM), median(Data.500.reps.SENS$WKNN))
Std.error.500.reps.SENS <- c(std.error(Data.500.reps.
   SENS$Logit), std.error(Data.500.reps.SENS$Probit),
   std.error(Data.500.reps.SENS$NN), std.error(Data
   .500.reps.SENS$SVM), std.error(Data.500.reps.SENS$
   WKNN))

#SPECIFICITY
names(Data.500.reps.SPEC)[names(Data.500.reps.SPEC) ==
   "results_LR.logit...3."] <- "Logit"
names(Data.500.reps.SPEC)[names(Data.500.reps.SPEC) ==
   "results_LR.probit...3."] <- "Probit"
names(Data.500.reps.SPEC)[names(Data.500.reps.SPEC) ==
   "results_nn...3."] <- "NN"
```

```
names(Data.500.reps.SPEC)[names(Data.500.reps.SPEC) ==
   "results_svm...3."] <- "SVM"
names(Data.500.reps.SPEC)[names(Data.500.reps.SPEC) ==
   "results_wknn...3."] <- "WKNN"

Mean.500.reps.SPEC <- c(mean(Data.500.reps.SPEC$Logit),
    mean(Data.500.reps.SPEC$Probit), mean(Data.500.reps
   .SPEC$NN), mean(Data.500.reps.SPEC$SVM), mean(Data
   .500.reps.SPEC$WKNN))
Median.500.reps.SPEC <- c(median(Data.500.reps.SPEC$
   Logit), median(Data.500.reps.SPEC$Probit), median(
   Data.500.reps.SPEC$NN), median(Data.500.reps.SPEC$
   SVM), median(Data.500.reps.SPEC$WKNN))
Std.error.500.reps.SPEC <- c(std.error(Data.500.reps.
   SPEC$Logit), std.error(Data.500.reps.SPEC$Probit),
   std.error(Data.500.reps.SPEC$NN), std.error(Data
   .500.reps.SPEC$SVM), std.error(Data.500.reps.SPEC$
   WKNN))



results.reps.500 <- format(round(data.frame(Mean.500.
   reps.ROC, Median.500.reps.ROC, Std.error.500.reps.
   ROC,
Mean.500.reps.SENS, Median.500.reps.SENS, Std.error
   .500.reps.SENS,
Mean.500.reps.SPEC, Median.500.reps.SPEC, Std.error
   .500.reps.SPEC),2),nsmall=2)
```

```
write.csv(results.reps.500, file = "results.reps
   .500.0.3.p.csv")
```

Skewed Predictors

```
#sample size
n=1000

#coefficients
#For average prob<=0.3
b0 <- -0.7
b1 <- -1
b2 <- -1
b3 <- -1

#For average prob>=0.5
#b0 <- 0.001
#b1 <- 0.003
#b2 <- 0.001
#b3 <- 0.02

#For average prob>=0.7
b0 <- 0.7
b1 <- 1
b2 <- 1
```

```
b3 <- 1

#randomly generating predictors
x1 <- rlnorm(n, 0, 1)
x2 <- rlnorm(n, 0, 1)
x3 <- rlnorm(n, 0, 1)

#hist(x1)

#randomly generating an error term
e <- rnorm(n,0,1)
xb <- b0 + b1*x1 + b2*x2 + b3*x3
prob <- 1/(1 + exp(-(xb+e)))
y <- rbinom(n, size = 1, prob = prob)
data <- data.frame(x1,x2,x3,y)
#refer to the "binning predictors" code for model
   fitting code
```

High/low Dimension

```
#sample size
n <- 100

#number of predictors
f=150
```

```
features <- replicate(f, expr= runif(n, 0, 1))
x <- as.matrix(features)
#b0 <- -0.7 #0.30 prob
#b <- as.matrix(runif(f,-0.2,0.2)) #0.30 prob
#b0 <- 0.001 #0.55 prob
#b <- as.matrix(runif(f,0.001,0.01)) #0.55 prob
b0 <- 0.1 #0.72 prob
b <- as.matrix(runif(f,0.001,0.03)) #0.70 prob
e <- rnorm(n,0,1) #error term
xb <- b0 + x%*%b
prob <- 1/(1 + exp(-(xb+e)))
summary(prob)
y <- rbinom(n, size = 1, prob = prob)
data <- data.frame(features,y)
#refer to the "binning predictors" code for model
   fitting code
```

Predictor Selection

```
#sample size
n <- 1000

#>=0.3
b0 <- -0.7
b1 <- -1
```

```
b2 <- -1
b3 <- -1
b4 <- 0.15
b5 <- 0.2
b6 <- -0.03
b7 <- -0.1
b8 <- 0.01
b9 <- 0.5
b10 <- 0.1

#=0.5
b0 <- 0.001
b1 <- 0.003
b2 <- 0.001
b3 <- 0.02
b4 <- 0.03
b5 <- 0.01
b6 <- 0.011
b7 <- 0.015
b8 <- 0.05
b9 <- 0.07
b10 <- 0.01

#>0.7
b0 <- 0.7
b1 <- 1
b2 <- 1
```

```
b3 <- 1
b4 <- 0.2
b5 <- 0.05
b6 <- -0.05
b7 <- 0.02
b8 <- 0.03
b9 <- 0.01
b10 <- -0.01

#randomly generating predictors
x1 <- runif(n,0,1)
x2 <- runif(n,0,1)
x3 <- runif(n,0,1)
x4 <- runif(n,0,1)
x5 <- runif(n,0,1)
x6 <- runif(n,0,1)
x7 <- runif(n,0,1)
x8 <- runif(n,0,1)
x9 <- runif(n,0,1)
x10 <- runif(n,0,1)

e <- rnorm(n,0,1) #error term

xb <- b0 + b1*data$x1 + b2*data$x2 + b3*data$x3+ b4*
   data$x4 + b5*data$x5 + b6*data$x6+ b7*data$x7 + b8*
   data$x8 + b9*data$x9+ b10*data$x10
```

```
prob <- 1/(1 + exp(-(xb+e)))
y <- rbinom(n, size = 1, prob = prob)
data <- data.frame(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,y)
#refer to the "binning predictors" code for model
   fitting code

#Feature selection

# define the control
control <- rfeControl(functions=caretFuncs, method="
   repeatedcv",  repeats = 50, number=10)

# Logit
lr.f <- rfe(rfeControl=control, data[,1:10], data$y,
   sizes=c(1:5), method="glm", family="binomial")
# summarize the results
print(lr.f)

#Probit
lr.probit.f <- rfe(rfeControl=control, data[,1:10],
   data$y, sizes=c(1:5), method="glm", family="binomial
   "(link = "probit"))
# summarize the results
print(lr.probit.f)

#NN
```

```
nn.f <- rfe(data[,1:10], data$y, sizes=c(1:5),
   rfeControl=control, method="nnet")
# summarize the results
print(nn.f)

#SVM
svm.f <- rfe(data[,1:10], data$y, sizes=c(1:5),
   rfeControl=control, method="svmLinear")
# summarize the results
print(svm.f)

#KKNN
wknn.f <- rfe(data[,1:10], data$y, sizes=c(1:5),
   rfeControl=control, method="kknn")
# summarize the results
print(wknn.f)
#During feature selection with balanced classes and 50
   repeats cross validation, the LR with both logit and
    probit link functions chose
#only two predictors (x8 and x9 respectively), SVM
   selected all predictors with x8,x9,x1,x3,x5 as the
   top 5, the NN selected 2 predictors (x8,x5)
#the WKNN selected only one predictor (x8). Therefore
   we will choose x8,x9, and x5 as predictors.

#After feature selection
```

```
data.f <- data[,-c(1:4,6,7,10)]
#refer to the "binning predictors" code for model
   fitting code
```

Bibliography

Beck, M. W. (2018). NeuralNetTools: Visualization and analysis tools for neural networks. *Journal of Statistical Software*, 85(11):1–20. 19

Boyacioglu, M. A., Kara, Y., and Baykan, Ö. K. (2009). Predicting bank financial failures using neural networks, support vector machines and multivariate statistical methods: A comparative analysis in the sample of savings deposit insurance fund (sdif) transferred banks in Turkey. *Expert Systems with Applications*, 36(2):3355–3366. 9

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159. 17

Bui, D. T., Tuan, T. A., Klempe, H., Pradhan, B., and Revhaug, I. (2016). Spatial prediction models for shallow landslide hazards: a comparative assessment of the efficacy of support vector machines, artificial neural networks, kernel logistic regression, and logistic model tree. *Landslides*, 13(2):361–378. 10

Byun, H. and Lee, S.-W. (2003). A survey on pattern recognition applications of support vector machines. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(3):459–486. 5

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46. 17

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297. 4

Doran, M., Raicu, D. S., Furst, J. D., Settimi, R., Schipma, M., and Chandler, D. P. (2007). Oligonucleotide microarray identification of bacillus anthracis strains using support vector machines. *Bioinformatics*, 23(4):487–492. 5

Dreiseitl, S. and Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*, 35(5-6):352–359. 4

Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(4):325–327. 4

Eftekhar, B., Mohammad, K., Ardebili, H. E., Ghodsi, M., and Ketabchi, E. (2005). Comparison of artificial neural network and logistic regression models for prediction of mortality in head trauma based on initial clinical data. *BMC Medical Informatics and Decision Making*, 5(1):1–8. 4

El_Jerjawi, N. S. and Abu-Naser, S. S. (2018). Diabetes prediction using artificial neural network. *International Journal of Advanced Science and Technology*, 121:55–64. 10

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422. 22

Hechenbichler, K. and Schliep, K. (2004). Weighted k-nearest-neighbor techniques and ordinal classification. `http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-1769-9`. Online; accessed 30 June 2020. 16

Huang, W., Nakamori, Y., and Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10):2513–2522. 5

Hunt, T. (2020). *ModelMetrics: Rapid Calculation of Model Metrics*. R package version 1.2.2.2. 19

Inayah, N., Lestari, I., and Musti, M. (2020). Comparative analysis of performance k-nearest neighbor and weighted k-nearest neighbor combined with global encoding in predicting protein-protein interaction human immunodeficiency virus type 1 with human. In *AIP Conference Proceedings*, volume 2242, page 030013, Depok, Indonesia. AIP Publishing LLC. 11

Intrator, O. and Intrator, N. (2001). Interpreting neural-network results: a simulation study. *Computational Statistics & Data Analysis*, 37(3):373–393. 5

Izenman, A. J. (2008). Modern multivariate statistical techniques. *Regression, Classification and Manifold Learning*, 10:978–0. 11, 14

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning*, volume 1398, pages 137–142, Berlin, Heidelberg. Springer. 14

Khan, J., Wei, J. S., Ringner, M., Saal, L. H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C. R., Peterson, C., et al. (2001). Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7(6):673–679. 8

Khuman, M. M. B. (2013). Classification of remote sensing data using k-nn method. *J. Inf. Knowl. Res. Electron. Commun. Eng*, 2:817–821. 15

Kim, Y. S. (2008). Comparison of the decision tree, artificial neural network, and linear regression methods based on the number and types of independent variables and sample size. *Expert Systems with Applications*, 34(2):1227–1234. 5

Kim, Y. S. (2010). Performance evaluation for classification methods: A comparative simulation study. *Expert Systems with Applications*, 37(3):2292–2306. 5, 6, 30, 31, 32

Kim, Y. S. (2011). Prediction of hotel bankruptcy using support vector machine, artificial neural network, logistic regression, and multivariate discriminant analysis. *The Service Industries Journal*, 31(3):441–468. 9

Kuhn, M. (2021). *caret: Classification and Regression Training*. R package version 6.0-90. 19

Kumar, U. A. (2005). Comparison of neural networks and regression analysis: A new insight. *Expert Systems with Applications*, 29(2):424–430. 8

Li, S., Wunsch, D. C., O'Hair, E., and Giesselmann, M. G. (2001). Comparative analysis of regression and artificial neural network models for wind turbine power curve estimation. *J. Sol. Energy Eng.*, 123(4):327–332. 5, 13

Lin, Y.-P., Chu, H.-J., Wu, C.-F., and Verburg, P. H. (2011). Predictive ability of logistic regression, auto-logistic regression and neural network models in empirical land-use change modeling–a case study. *International Journal of Geographical Information Science*, 25(1):65–87. 9

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133. 4

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2021). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.7-9. 19

Min, J. H. and Lee, Y.-C. (2005). Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. *Expert Systems with Applications*, 28(4):603–614. 5, 8

Mohsen, H., El-Dahshan, E.-S. A., El-Horbaty, E.-S. M., and Salem, A.-B. M. (2018). Classification using deep learning neural networks for brain tumors. *Future Computing and Informatics Journal*, 3(1):68–71. 10, 13

Morris, T. P., White, I. R., and Crowther, M. J. (2019). Using simulation studies to evaluate statistical methods. *Statistics in Medicine*, 38(11):2074–2102. 20

Muniz, A., Liu, H., Lyons, K., Pahwa, R., Liu, W., Nobre, F., and Nadal, J. (2010). Comparison among probabilistic neural network, support vector machine and logistic regression for evaluating the effect of subthalamic stimulation in Parkinson disease on ground reaction force during gait. *Journal of Biomechanics*, 43(4):720–726. 9

Musa, A. B. (2013). Comparative study on classification performance between support vector machine and logistic regression. *International Journal of Machine Learning and Cybernetics*, 4(1):13–24. 4, 9

Pearson, K. (1904). *On the theory of contingency and its relation to association and normal correlation*. Cambridge University Press, London. 16

Portela González, J., Muñoz San Roque, A., and Pizarroso Gonzalo, J. (2020). *NeuralSens: Sensitivity Analysis of Neural Networks*. R package version 0.2.2. 19

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. 19

Rizwan, M. and Anderson, D. V. (2016). Comparison of distance metrics for phoneme classification based on deep neural network features and weighted k-nn classifier. In *Workshop on Machine Learning in Speech and Language Processing*, San Francisco. 5

Sadri, J., Suen, C. Y., and Bui, T. D. (2003). Application of support vector machines for recognition of handwritten Arabic/Persian digits. In *Proceedings of Second Iranian Conference on Machine Vision and Image Processing*, volume 1, pages 300–307, Tehran. 14

Salas-Eljatib, C., Fuentes-Ramirez, A., Gregoire, T. G., Altamirano, A., and Yaitul, V. (2018). A study on the effects of unbalanced data when fitting logistic regression models in ecology. *Ecological Indicators*, 85:502–508. 10

Schliep, K. and Hechenbichler, K. (2016). *kknn: Weighted k-Nearest Neighbors*. R package version 1.3.1. 19

Stylianou, N., Akbarov, A., Kontopantelis, E., Buchan, I., and Dunn, K. W. (2015). Mortality risk prediction in burn injury: Comparison of logistic regression with machine learning approaches. *Burns*, 41(5):925–934. 10

SubbaNarasimha, P., Arinze, B., and Anandarajan, M. (2000). The predictive accuracy of artificial neural networks and multiple regression in the case of skewed data: Exploration of some issues. *Expert Systems with Applications*, 19(2):117–123. 8

Tay, F. E. and Cao, L. (2001). Application of support vector machines in financial time series forecasting. *Omega*, 29(4):309–317. 14

Tu, J. V. (1996). Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of Clinical Epidemiology*, 49(11):1225–1231. 4

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. ISBN 0-387-95457-0. 19

Wang, Z., Na, J., and Zheng, B. (2020). An improved kNN classifier for epilepsy diagnosis. *IEEE Access*, 8:100022–100030. 11

Wickham, H., Girlich, M., and Ruiz, E. (2021). *dbplyr: A 'dplyr' Back End for Databases*. R package version 2.1.1. 19

Zhao, M. and Chen, J. (2016). Improvement and comparison of weighted k nearest neighbors classifiers for model selection. *Journal of Software Engineering*, 10(1):109–118. 5, 15