EVALUATING CREDIT SCORING MODELS

Vesarach Aumeboonsuke

A Dissertation Submitted in Partial

Fulfillment of the Requirements for the Degree of

Doctor of Philosophy (Finance)

School of Business Administration

National Institute of Development Administration

2011

EVALUATING CREDIT SCORING MODELS

Vesarach Aumeboonsuke

School of Business Administration

Associate Professor ……[illegible]…………….Major Advisor

(Arthur Lance Dryver, Ph.D.)

The Examining Committee Approved This Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Finance).

Assistant Professor ……[illegible]……………..Committee Chairperson

(Viput Ongsakul, Ph.D.)

Assistant Professor ……[illegible]………………..Committee

(Sarayut Nathaphan, Ph.D.)

Associate Professor ……[illegible]………………...Committee

(Arthur Lance Dryver, Ph.D.)

Associate Professor ……[illegible]…………………Dean

(Boonchai Hongcharu, Ph.D.)

April 2012

ABSTRACT

Title of Dissertation	Evaluating Credit Scoring Models
Author	Miss Vesarach Aumeboonsuke
Degree	Doctor of Philosophy (Finance)
Year	2011

Evaluating the credit worthiness of credit seekers is a crucial process for financial institutions simply because their existence largely depends on how such a process is conducted. Financial institutions use a variety of credit scoring methods and a variety of criteria to select the best credit scoring methods. The primary purpose of this research is to evaluate the performance of some of the existing popular credit scoring methods that are widely used by financial institutions. The credit scoring methods to be considered for comparison purpose include logistic regression, discriminant analysis, and recursive partitioning. Several statistical criteria to be considered for evaluation include the Kolmogorov-Smirnov statistic (K-S), the Gini coefficient, and odds ratio at various cut-offs.

Much research in the past has compared credit scoring methods through using sets of real-world data. In this paper, however, the comparison of the credit scoring methods has been done by using a set of data generated through simulation in order to acquire extensively representative and sufficiently effective samples; in this way, it is possible to compare and validate the performance of the classification models on the population.

This paper simulates the data sets of the population, draw samples from each population set, runs each credit scoring method on each data set, computes the K-S, Gini, and odds ratio for each model for each data set, compares the cross-validation with the K-S, Gini, and odds ratio at various cut-off points, and evaluates the performance of different credit scoring models across different methods, across samples with different ratios of "goods" to "bads", and across samples drawn from the population with different characteristics.

The findings of this research will be useful for financial institutions, especially commercial banks, because they present evidence of how well each credit scoring method can predict the credit score of loan applicants. Banks make lending decisions based on such credit scoring systems, and the lending decision is crucial because it is the source of their revenue. If the bank accepts the applicant that is going into default, then it will have a bad loan, which results in loan losses. On the other hand, if the bank rejects the applicant that is not going into default, then the bank has lost the opportunity to gain more revenue from that applicant. Therefore, ideally, banks would like to use a credit scoring model that has a stable and reliable predictive power across different characteristics of populations and sample sets.

ACKNOWLEDGEMENTS

First and foremost I offer my sincerest gratitude to my advisor, Associate Professor Dr. Arthur Lance Dryver, for his encouragement, advice, and support. Without him this thesis would not have been completed or written. One simply could not wish for a better or friendlier advisor.

I am grateful to my family especially my parents, Kasem Aumeboonsuke and Ma. Laarni C. Aumeboonsuke, for their love and encouragements throughout my life.

I would like to thank my teachers at Assumption University, especially Dr. Cherdpong Siboonruang, for his motivation and guidance. I remember the generosity and inspiration of Dr. Tareque Nasser and Dr. Wiyada Nittayakasetwat, who introduced me to the fascinating world of finance. I would also like to thank Dr. Pathathai Sinliamtong for being a very supportive and understanding supervisor.

I would like to acknowledge the financial and academic support of the NIDA Business School, particularly in the award of a full Ph.D. scholarship. I would like to thank the committee members, Asst. Prof. Dr. Viput Ongsakul and Asst. Prof. Dr. Sarayut Nathaphan for their constructive criticism and insightful comments.

I have been aided by Danai Likitratcharoen, my senior in the Ph.D. program, for many months in running the R codes. I would also like to thank Nopphon Tangjitprom for his friendship and support; in many ways I have learnt much from and because of him.

I am also indebted to the contributors to the "Open Source" R programming community for providing the numerous tools and systems I have used to produce my thesis results.

My sincere thanks are due to the wonderful editor, Dr. Bruce Leeds, for his detailed review and excellent advice.

Lastly, I offer my regards and blessings to my teachers and all of those who supported me in any respect during the completion of this thesis.

Vesarach Aumeboonsuke

April 2012

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Credit scoring is the set of decision models and their underlying techniques that aid lenders in the granting of consumer credit (Thomas, Edelman and Crook, 2002). The ontology of the credit scoring model is illustrated in Figure 1. According to Figure 1, many loan applicants approach a bank to request loans, and these applicants are required to submit information such as age, gender, employment, residential status, number of dependents, etc. to the bank. Consequently, the bank will use the credit scoring model to process the information and compute the credit score of each loan applicant. The credit score is an interval scale that varies from zero to any number depending on the user's predetermined range. It can be scaled to range from zero to one, so that it represents the probability that the loan applicant will not default on the loan. The bank will set the critical credit score as a benchmark such that, the bank will accept any loan applicant whose score falls above the critical score. On the other hand, the bank will reject any loan applicant whose score falls between zero and the critical score. In addition, the bank may perform further manual intervention, thus overriding the credit score results.

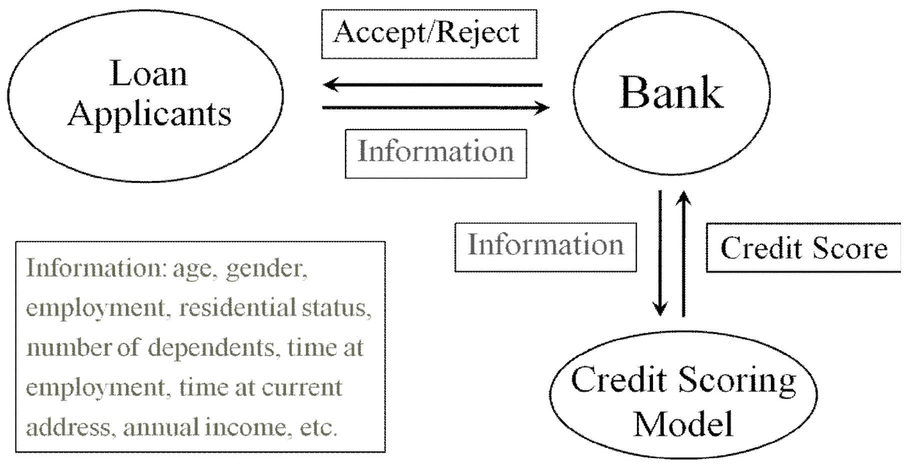
Accept/Reject
Loan Applicants
Bank
Information
Information
Credit Score
Information: age, gender, employment, residential status, number of dependents, time at employment, time at current address, annual income, etc.
Credit Scoring Model

Figure 1.1 The Ontology of the Credit Scoring Model

Evaluating the credit worthiness of credit seekers is a crucial process for financial institutions simply because their existence largely depends on how such a process is conducted. Financial institutions use a variety of credit scoring methods and a variety of criteria to select the best credit scoring methods. The primary purpose of this research is to evaluate the performance of some of the existing popular credit scoring methods that are widely used by financial institutions. The credit scoring methods to be considered for comparison purposes include the following:

1) Logistic regression
2) Discriminant analysis
3) Recursive partitioning

The criterion to be used for comparing the performances of these methods is their predictability or explanatory power, which is the number of times a given method identifies correctly credit-worthy customers, or correctly rejects non-credit-worthy customers. Thomas, Edelman and Crook (2002) have suggested several statistical criteria to be considered for evaluation purposes including:

1) Kolmogorov-Smirnov statistic (K-S)
2) Gini coefficient (Gini)
3) Odds ratio

Much of the past research has compared credit scoring methods by using sets of real-world data. In this paper, however, the comparison of the credit scoring methods has been done by using a set of data generated through simulation. The reasons for using this set of simulated data are as follows: first, it is illegal in some countries for banks to provide the personal information of their clients to outsiders, so researchers are unable to obtain real data from the banks. The second reason for generating data is to acquire extensively representative and sufficiently effective samples because the purpose of this research is to compare and validate the performance of the classifications models so population data are needed in order to validate the models estimated from the samples and then to test the models with reference to the population. Finally, real-world data creates a biased comparison because it contains two sets of data; the first set contains observations that are accepted by the banks, but the second set of data contains observations that are rejected by the banks. The first set of data can be classified into good loans when the person pays the loan back and a bad loan when the person defaults; however, the second set of data cannot be classified because the person has not been accepted in the first place. As a result, it is impossible to know whether each observation in the second set of data falls into the good loan group or bad loan group. In this case, whether the observation in the second set of data is in the category of either the good loan group or the bad loan group is left unknown. For these reasons, it is more appropriate to use synthetic data.

The findings of this research will be useful for financial institutions, especially commercial banks, because evidence is presented concerning how well each credit scoring method can predict the credit score of the loan applicant. Banks make lending decisions based on such credit scoring systems, and the lending decision is crucial because it is the source of their revenue (Altman, 1980; Mester, 1997; Atiya, 2001). If the bank accepts the applicant that is going into default, then the bank will have a bad loan, which results in loan losses. Contrariwise, if the bank rejects the applicant that is not going into default, then the bank has lost the opportunity to gain more revenue from that applicant. In this case, the bank experiences an opportunity cost. Therefore, ideally, the banks would like to use a credit scoring model that can predict or distinguish good loan applicants from bad loan applicants.

CHAPTER 2

CREDIT SCORING IN THE LITERATURES

There are many credit scoring methods that have been proposed and used in the pertinent literature, for example, multiple linear regression (Hand and Henley, 1996, 1997; Mayers and Forgy, 1963; Orgler, 1970), discriminant analysis (Abdou, Pointon and El-Masry, 2008; Desai, Crook and Overstreet, 1996; Eisenbeis, 1987; Hand and Henley, 1996, 1997; Kolesar and Showers, 1985; Mayers and Forgy 1963; Orgler 1970; Press and Wilson, 1978; Reichert, Cho and Wagner, 1983; Ripley, 1994; Rosenberg and Gleit, 1994; Srinivasan and Kim, 1987; West, 2000; Wiginton, 1980), logistic regression (Abdou, Point and El-Masry, 2008; Desai, Crook and Overstreet, 1996; Eisenbeis, 1987; Galindo and Tamayo, 2000; Hand and Henley, 1996, 1997; Press and Wilson, 1978; Srinivasan and Kim, 1987; West, 2000; Wiginton, 1980), neural networks (Abdou, Point and El-Masry, 2008; Baesens et al., 2003; Desai, Crook and Overstreet, 1996; Galindo and Tamayo, 2000; Hand and Henley, 1996, 1997; Hsieh, 2005; Ripley, 1994; West, 2000), data envelopment analysis (Emel et al., 2003), and some non-parametric methods such as the k-nearest neighbor (Galindo and Tamayo, 2000; Hand and Henley, 1996, 1997), and the decision tree (recursive partitioning) (Galindo and Tamayo, 2000; Hand and Henley, 1996, 1997; Rosenberg and Gleit, 1994).

Discriminant analysis was first introduced to credit scoring by Durand (1941), followed by Mayers and Forgy (1963), who produced evidence that the performance of discriminant analysis was better compared to stepwise multiple linear regression analysis, and Wiginton (1980), who found that logistic regression gave superior predictability when compared with discriminant analysis.

Ripley (1994) and Rosenberg and Gleit (1994) introduced the applications of neural networks to credit decisions and fraud detection. Later on, many researchers employed neural networks in their studies. The study of Hand and Henley (1996, 1997), for example, suggested that the K-nearest neighbor was the best model, while Galindo and Tamayo (2000) found that the decision tree was the best model. However, West (2000) showed that neural networks had less predictive power when compared with the traditional methods in general. In addition, West (2000) also suggested that logistic regression was superior to neural networks in particular. On the other hand, Abdou, Point and El-Masry (2008) found that the neural networks approach was superior to other models, such as logistic regression and discriminant analysis.

The contradicting findings offered by Hand and Henley (1996, 1997), Galindo and Tamayo (2000), West (2000) and Abdou, Point and El-Masry (2008) put these approaches as to which one was superior, suggesting the need for more in-depth study in order to reach decisive conclusions. Table 2.1 offers conclusions regarding the contradictory findings of the four studies mentioned above. According to Table 2.1, these contradictory results may come from using different data sets, different sample sizes, different ratios of bad-to-good, or different numbers of independent variables.

Table 2.1 The Contradictory Findings in Previous Literature

Literature	**Best model**	**Data**	**Sample Size**	**Good: Bad**	**# of X**
Hand & Henley (1996, 1997)	K-nearest neighbor	Mail order company, UK	15,054 4,132	54.5 : 45.5 54.7 : 45.3	16
Galindo & Tamayo (2000)	Decision tree	Home mortgage loans from one financial institution, Mexico	4,000	50.8 : 49.2	24
West (2000)	Logistic regression	German data, Australian data	1,000 690	70 : 30 44.5 : 55.5	14
Abdou et al. (2008)	Neural network	4-year personal loans from one bank, Egypt	581	74.5 : 25.5	20

This study would like to reconcile the contradictory evidence from previous studies concerning the performance of each model. In order to achieve this purpose, population sets with different ratios of good loans to bad loans are formed, and various sample groups are drawn from each population with different ratios of good loans to bad loans. In this way, the research is able to test if the ratio of good loans to bad loans and the sample size will affect the performance of each model.

Many researchers in the past, as mentioned above, used real-world data. However the real data used in these studies reflect some bias in the sense that the sample set of data contains the observations that are accepted by the banks but does not contain the observations that are rejected by the banks. Wiginton (1980) has stated that "…The data were presensored. That is, the credit applications had already been screened by credit officers who rejected the 'bad risks,' thus removing much of the variability of interest in the data and leaving only the pathological cases, which would tend to confound any modeling effort…." This argument is presented in Figure 2.1, where the credit scoring model is formed based on the sample set, including "bad accepted" and "good accepted," but excluding "bad rejected" and "good rejected." However, the "bad rejected" group is the most crucial group to include in the model because the purpose of the model is to correctly identify bad borrowers so that the bank does not have a bad loan.

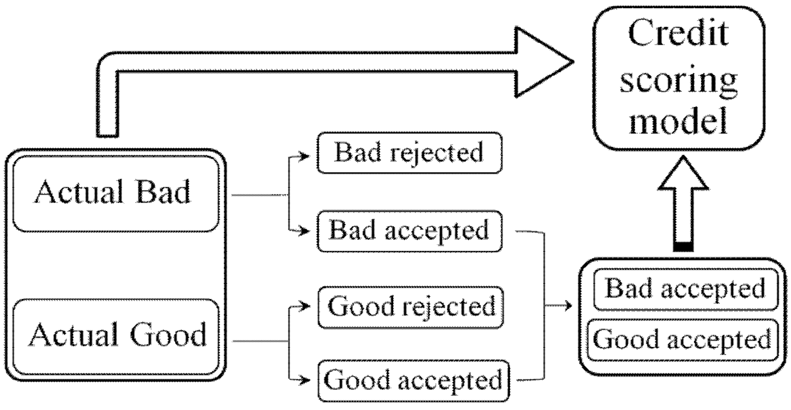

Figure 2.1 The Argument for Using Synthetic Data.

Therefore, the researcher of this study believes that using real-world data to evaluate the models of interest yields biased results. Moreover, the real-world data represent the sample set. This research would like to assess how well each classification model performs using the sample set as a training set and using the population set as a test set. However, the real-world data do not contain enough information about the population. In order to overcome this disadvantage, the simulation technique is used to generate the data used in evaluating the credit scoring models. Some simulations related to credit scoring models have been produced by Dryver (2011) and Dryver and Jantra Sukkasem (2009). Dryver (2011) simulated two populations and showed that the K-S statistic can be used for model comparison and selection but it may lack sensitivity in some cases. Dryver and Jantra Sukkasem (2009) simulated three populations and used logistic regression to investigate the precision of the K-S statistic, the Gini coefficient, and odds at various cut-off points.

CHAPTER 3

CREDIT SCORING METHODS

Many credit scoring methods have been proposed and used in the literature, as mentioned in the literature review. However, this research concentrates on the following methods:

1) Logistic Regression
2) Discriminant Analysis
3) Recursive Partitioning

One important requirement for a practical and legitimate credit scoring model is that the model can also offer the reason why it rejects or accepts loan applicants. This kind of model feature enables banks to provide support that justifies their decisions regarding loan applications. As a result, although there are many statistical models that can predict y-value as “yes” or “no,”: this research only uses the models that satisfy the requirement mentioned above.

According to Galindo and Tamayo (2000), another important issue for financial decision making is the transparency or degree of interpretability of models. Transparent models are those that are conceptually understood by the decision maker, such as a decision tree expressed in term of profiles or rule sets. By contrast, while neural networks can act as accurate black boxes, they are opaque and not able to provide simple clues about the basis for their classifications or predictions. Elder and Pregibon (1996) argue that if accuracy is acceptable, a more interpretable model is more useful than a “black box.”

Moreover, it is feasible to use multiple linear regression, logistic regression, and discriminant analysis to predict the credit score and to rank the applicants based on their scores without any additional adjusting procedure to make a meaningful comparison. Apart from these three models, recursive partitioning (decision tree) can categorize the observations into many groups based on the credit scores. Although the observations within the same group are given equal credit scores, each group is given a different credit score so it is feasible to rank different groups of observations. However, the k nearest neighbor can only estimate the predicted y as "yes" (accept) or "no" (reject). In this case, it is impossible to use these two models to rank the applicants. In addition, if there are two classes for the dependent variables, multiple linear regression and discriminant analysis will produce identical results (Flury and Riedwyl, 1985; Lawrence et al., 2010).

As a result of this observation, this research only focuses on logistic regression, discriminant analysis, and recursive partitioning where it is transparent, not redundant, and possible to rank the observations based on their predicted probability of being a credit worthy client, so it is reasonable to use the K-S, Gini, and odds ratio to compare and contrast the models.

In addition to the K-S, Gini, and odds ratio, the research also validates the models by conducting cross validation at various cut-off scores in order to determine the percent at which the model predicts correctly (that is the predicted y = yes is the same as the actual y = not default and the predicted y = no is the same as the actual y = default). In other words, the cross validation reports the type I and type II errors of each model. Table 3.1 presents information on the cross validation, where the good model is the model that correctly classifies each loan applicant and minimizes type I and type II errors.

Table 3.1 Cross Validation

Data Set		**Actual**	
		Bad	**Good**
Predicted	Bad	Correct Assessment	Type II Error
	Good	Type I Error	Correct Assessment

3.1 Logistic Regression

Logistic regression (logit) is used for predicting the probability of the occurrence of an event by fitting data into a logistic curve. It is a generalized linear model used for binomial regression. Similar to linear regression analysis, it makes use of several predictor variables that may be either numerical or categorical (Hosmer and Lemeshow, 2000). For example, the probability that a person will default on his or her bank loan might be predicted from the person's marital status (categorical, for example, "single" = class "1," "married" = class "2"), and number of dependents (numerical, for example, 0, 1, 2, 3, etc.).

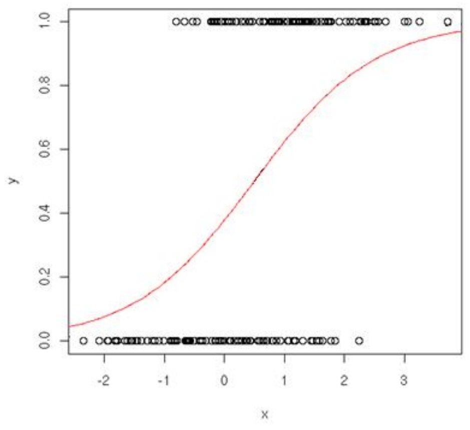

Figure 3.1 Logistic Function

$$f(z) = \frac{e^z}{e^z+1} = \frac{1}{1+e^{-z}} \tag{1}$$

where

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k \tag{2}$$

A graph of the logistic function is shown in Figure 3.1 The logistic function is defined as equation (1) and the variable z is usually defined as equation (2), where β_0 is the intercept and β_1, β_2, β_3, …,β_k are the regression coefficients of x_1, x_2, x_3,…,x_k, respectively. The intercept is the value of z when the value of all independent variables is zero (e.g. the value of z is someone with no risk factors). Each of the regression coefficients describes the weight or size of the contribution of that factor. In equation (2), the sum of the product between each explanatory variable and its weight is "z," which is similar to the predicted value from multiple linear regression. Then, the "z" is substituted into equation (1) to compute "f(z)," which is the predicted value for the logistic function. The logistic function is practical in terms of predicting the probability because it can take any value of explanatory variables from negative infinity to positive infinity, whereas the predicted value “f(z)” is limited to the range of zero and one, which makes it reasonable to represent the value of the probability of a particular outcome.

A positive regression coefficient means that that explanatory variable increases the probability of the outcome, while a negative regression coefficient means that the variable decreases the probability of that outcome; a large regression coefficient means that the risk factor strongly influences the probability of that outcome, while a near-zero regression coefficient means that that risk factor has little influence on the probability of that outcome.

Logistic regression is a useful way of describing the relationship between one or more independent variables (e.g., age, sex, etc.) and a binary response variable that has only two possible values such as defaulting on a loan (the observation has a response variable equal to “0” if that observation is the person that “defaults,” and the observation has a response variable equal to “1” if that observation is the person that “does not default”).

According to equation (1), the logistic function has the predicted value (or predicted credit score) f(z), which is expressed as the predicted probability that a person will “not default,” and f(z) can take any value in the range of [0,1]. In order to utilize the predicted value from the logistic function to make a proper loan approval decision, the bank will set the cut-off score “c,” which serves as the critical value to classify the applicants as “bad” or “good.” In this case, the loan applicant whose predicted credit score f(z) falls into the range of [0,c] will be given a response variable equal to “0” (will default) and the bank will reject the loan application. On the other hand, the loan applicant whose predicted credit score f(z) falls into the range of (c,1] will be given a response variable equal to “1” (will not default) and the bank will approve the loan.

3.2 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is the statistical method for finding a linear combination of features which characterize or separate two or more mutually-exclusive and exhaustive classes of objects or events based on a set of measurable object’s features.

LDA is closely related to ANOVA (analysis of variance) and linear regression analysis, which also attempt to express one dependent variable as a linear combination of a set of independent variables. (Fisher, 1936) In the ANOVA and regression analysis, the dependent variable is a numerical quantity, while for LDA it is a categorical variable. Logistic regression is similar to LDA in the sense that the dependent variable is also a categorical variable. In LDA, the dependent variable (Y) is the group and the independent variables (X) are the object features that might describe the group. The dependent variable is always a category (nominal scale) variable while the independent variables can be any measurement scale (i.e. nominal, ordinal, interval, or ratio). LDA assumes that the groups can be separated by a linear combination of features that describe the objects.

The objective of LDA is to minimize total error of classification. In order to achieve this objective, the classification rule is to assign an object to the class with the highest conditional probability. For example, the object will be classified in the "good" class if, given a set of applicant features (x), the probability that an object belongs to the "good" class is higher that the probability that an observation belongs to the "bad" class; that is, $P(good|x) > P(bad|x)$.

The bank would like to know $P(good|x)$, which is unknown: however, the value of $P(x|good)$, the probability that an observation belongs to a particular set of features (x) given that the observation comes from a "good" class is known. According the Bayes Theorem, there is a relationship between $P(good|x)$ and $P(x|good)$, as shown in equation (6).

$$P(good|x) = \frac{P(x|good)\times P(good)}{P(x|good)\times P(good)+P(x|bad)\times P(bad)} \tag{6}$$

where *P(good)* = *Ng* / *N* = the probability that the observation is in the "good" class,

P(bad) = *Nb* / *N* = the probability that the observation is in the "bad" class,

Ng = the number of observations in the "good" class,

Nb = the number of observations in the "bad" class, and

N = *Ng* + *Nb* = the number of total observations.

However, to use the Bayes rule directly is impractical because in order to obtain $P(x|good)$ and $P(x|bad)$ so much data are acquired to obtain the relative frequencies of each group for each measurement. It is more practical to assume the distribution and obtain the probability theoretically. If we assume that each class has multivariate normal distribution and each class has the same covariance matrix, we can obtain the Linear Discriminant Analysis (LDA) function (Friedman, 1989; Hastie and Tibshirani, 1996).

The following example shows how we can obtain the LDA function and how LDA can be applied to credit scoring. Consider the case in which the bank would like to know whether a loan applicant is good (will pay back on time) or bad (will not pay back on time) based on several measurements of the loan applicant, such as age, average monthly income, average monthly spending, marital status, etc. The object (observation) is a loan applicant. Each measurement of the loan applicant is the features that describe the object. These several features serve as the independent variables. The class category "good" and "bad" of the loan applicant is what the bank is looking for. This class category is the dependent variable. Assume that there are two data sets, as shown in equation (3) and equation (4). Data set **A**, the "good" borrowers, consists of "p" observations, while data set **B**, the "bad" borrowers, consists of "q" observations, and a set of features contains "k" features (i.e. there are "k" explanatory factors, or independent variables).

$$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ a_{21} & \cdots & a_{2k} \\ \vdots & \ddots & \vdots \\ a_{p1} & \cdots & a_{pk} \end{bmatrix} \tag{3}$$

$$\mathbf{B} = \begin{bmatrix} b_{11} & \cdots & b_{1k} \\ b_{21} & \cdots & b_{2k} \\ \vdots & \ddots & \vdots \\ b_{q1} & \cdots & b_{qk} \end{bmatrix} \tag{4}$$

From equation (3) and equation (4), we can find the mean of each feature in each class, as shown in equation (5) and equation (6), and the global mean (mean of each feature based on the whole data set is presented in equation (7).

$$\boldsymbol{\mu_A} = [\mu_{a1} \quad \cdots \quad \mu_{ak}] \tag{5}$$

$$\boldsymbol{\mu_B} = [\mu_{b1} \quad \cdots \quad \mu_{bk}] \tag{6}$$

$$\boldsymbol{\mu} = [\mu_1 \quad \cdots \quad \mu_k] \tag{7}$$

where

$$\mu_{ak} = \frac{\sum_{i=1}^{p} a_{ik}}{p} \quad , \quad \mu_{bk} = \frac{\sum_{i=1}^{q} b_{ik}}{q} \quad , and \quad \mu_k = \frac{\sum_{i=1}^{p} a_{ik} + \sum_{i=1}^{q} b_{ik}}{p+q}$$

Equation (8) and equation (9) represent the deviation between the actual value of each feature in each observation and its mean value.

$$\boldsymbol{A^0} = \begin{bmatrix} a_{11} - \mu_1 & \cdots & a_{1k} - \mu_k \\ a_{21} - \mu_1 & \cdots & a_{2k} - \mu_k \\ \vdots & \ddots & \vdots \\ a_{p1} - \mu_1 & \cdots & a_{pk} - \mu_k \end{bmatrix} \tag{8}$$

$$\boldsymbol{B^0} = \begin{bmatrix} b_{11} - \mu_1 & \cdots & b_{1k} - \mu_k \\ b_{21} - \mu_1 & \cdots & b_{2k} - \mu_k \\ \vdots & \ddots & \vdots \\ b_{q1} - \mu_1 & \cdots & b_{qk} - \mu_k \end{bmatrix} \tag{9}$$

$$\boldsymbol{c_{good}} = \frac{(\boldsymbol{A^0})^T \ \boldsymbol{A^0}}{p} \tag{10}$$

$$\boldsymbol{c_{bad}} = \frac{(\boldsymbol{B^0})^T \ \boldsymbol{B^0}}{q} \tag{11}$$

According to equation (10), $\boldsymbol{c_{good}}$, the k x k matrix, is the covariance matrix of the data set in the class "good." Similarly, in equation (11), $\boldsymbol{c_{bad}}$, the matrix with the same dimension as $\boldsymbol{c_{good}}$, is the covariance matrix of the data set in the class "bad."

Based on equation (10) and equation (11), we can find $\boldsymbol{C}$, the covariance matrix of the whole data set, as shown in equation (12).

$$\boldsymbol{C_{(r,s)}} = \frac{1}{(p+q)} [p \cdot \boldsymbol{c_{good(r,s)}} + q \cdot \boldsymbol{c_{bad(r,s)}}] \tag{12}$$

where

$\boldsymbol{C_{(r,s)}}$ is the element in row "r" and column "s" of the covariance matrix $\boldsymbol{C}$,

$\boldsymbol{c_{good(r,s)}}$ is the element in row "r" and column "s" of the covariance matrix $\boldsymbol{c_{good}}$, and $\boldsymbol{c_{bad(r,s)}}$ is the element in row "r" and column "s" of the covariance matrix $\boldsymbol{c_{bad}}$.

The final step is to compute the Linear Discriminant Function (f_i) of each pre-specified class "i," Based on our example, there are two classes ("good" and "bad") so there will be two values of f_i; $f_{good}$and f_{bad}, as shown in equation (13) and equation (14).

$$f_{good} = \mu_{good} C^{-1} x_n^T - \frac{1}{2} \mu_{good} C^{-1} \mu_{good}^T + \ln(p) \tag{13}$$

$$f_{bad} = \mu_{bad} C^{-1} x_n^T - \frac{1}{2} \mu_{bad} C^{-1} \mu_{bad}^T + \ln(q) \tag{14}$$

where $x_n = [x_{n1} \quad \cdots \quad x_{nk}]$, $x_n$is the matrix showing the values of features that belong to object (or observation) *"n."* For example $x_{n1}$is the value of feature "1" of observation *"n."*

In order to decide whether loan applicant (observation) *"n"* falls into the "good" or "bad" class, the bank will compare the value of f_{good} and f_{bad} . The rule is to assign observation *"n"* to class *"i"* that has a maximum value of f_i.

In a special case where there are two classes for the dependent variables, Flury and Riedwyl (1985) and Lawrence et al. (2010) showed that the linear discriminant analysis will be mathematically equivalent to the multiple linear regression.

3.2.1 Multiple Linear Regression

Multiple linear regression attempts to model the relationship between two or more explanatory variables (independent) and a response variable (dependent) by fitting a linear equation with the observed data. Every value of the independent variable *x* is associated with the value of the dependent variable *y*. The regression line for *p* number of explanatory variables $x_1, x_2, \ldots, x_p$ is expressed by equation (15).

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i \quad \textit{for } i = 1, \ldots, n \tag{15}$$

These *n* equations can be written in vector form as shown by equation (16).

$$\boldsymbol{Y} = \boldsymbol{\beta X} + \varepsilon \tag{16}$$

where

$$\boldsymbol{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} x_1' \\ x_2' \\ \vdots \\ x_n' \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}, \quad \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

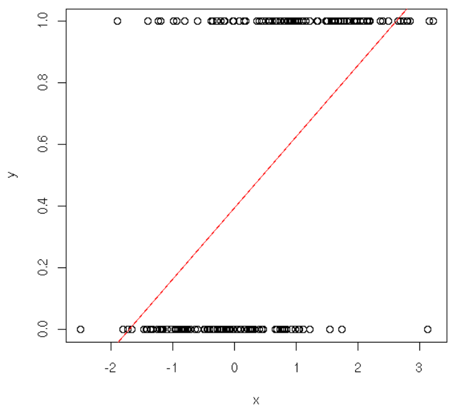

Figure 3.2 Linear Regression Function

Figure 3.2 shows the best-fitting line of the linear regression function. In the linear regression model, the best-fitting line for the observed data is calculated by minimizing the sum of the squares of the vertical deviations from each data point to the line. Because the deviations are squared, then summed, there are no cancellations between positive and negative values. The least-squares estimates b_0, b_1, ... b_p are the values fit by the equation (17) and residuals ε_i are equal to the difference between the observed and fitted values. The sum of the residuals is equal to zero. Multiple linear regression can be used to model the credit scores, as can be seen from Mayers and Forgy (1963) and Orgler (1970). Normally in practice, when banks use linear regression, the predicted value represents the credit score that can take the value from negative infinity to positive infinity and can be scaled to take the value from zero to one. The banks have their own cut-off score to determine if the applicant's score will pass or fail.

$$\hat{Y}_i = b_0 + b_1 x_{i1} + b_2 x_{i2} + \cdots + b_p x_{ip} \qquad (17)$$

where $\hat{Y}_i$ is the fitted value of observation i

x_{ij} is the value of the independent variable j for observation i

b_j is the estimated coefficient of each independent variable j

3.3 Recursive Partitioning

Recursive partitioning is a non parametric statistical method developed by Breiman and Friedman in 1973. This method creates a decision tree that strives to correctly classify members of the population based on binary dependent variables (Breiman et al., 1984). It has been widely applied for classification in many scientific fields such as biomedical field (Goldman et al., 1996, 1988; Zhang et al., 2001), engineering (Bahl et al., 1989), astronomy (Owens, Griffiths and Ratnatunga, 1996), and chemistry (Chen, Rusinko and Young, 1998). Recursive partitioning has an advantage that it has the feature of transparency; it can be represented as a set of rules in almost plain English. This makes it ideal for economic and financial applications.

The main characteristic of recursive partitioning is that the feature space, i.e. the space spanned by all predictor variables, is recursively partitioned into a set of rectangular areas. The partition is created such that observations with similar response values are grouped. After the partition is completed, a constant value of the response variable is predicted within each area. The partition produces a decision tree (classification tree) that expresses a sequential classification process in which a case (described by a set of attributes) is assigned to one of a disjoint set of classes. Each leaf of a tree denotes a class (Quinlan, 1987).

The rationale of the decision tree can be explained in more detail by means of a credit scoring example, as shown in . This example was taken from a book by Thomas, Edelman and Crook (2002). Given information on the set of the determinants of the credit quality of the loan applicants, the bank's aim is to predict whether the loan applicants will turn out to be "good" or "bad" (binary response variable) from a set of binary explanatory factors (whether the residential status of the loan applicant is that of an owner or not, whether the loan applicant has been a customer at the bank for more or less than two years, whether the loan applicant has children or not, whether the employment of the loan applicant is professional or not, whether the age of the loan applicant is more or less than 26 and 21 years, and if the loan applicant is not the owner of the house, whether the loan applicant lives with parents or not).

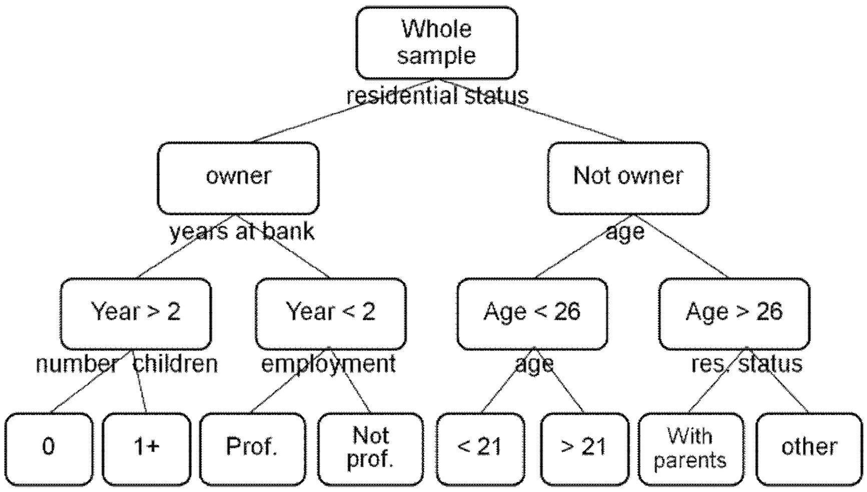

Figure 3.3 Recursive Partitioning (Decision Tree)

Figure 3.3 illustrates a decision tree that can be obtained from using the recursive partitioning model to classify the loan applicants into different groups. The set of loan application data is first split into two subsets, so that looking at the sample of previous applicants, these two new subsets of application attributes are far more homogeneous regarding the default risk of the applicants than the original set. Each of these sets is then again split into two to produce even more homogeneous subsets, and the process is repeated. This is why the approach is called recursive partitioning. The process stops when the subsets meet the requirements to be terminal nodes of the tree. Each terminal node is then classified as a “good” or “bad” borrower and the whole procedure can be presented graphically as the tree in. Figure 3.3.

Three decisions make up the classification tree procedure:

1) How to assign terminal nodes into good and bad categories
2) What rule to use to split the sets into two – the splitting rule
3) How to decide that a set is a terminal node – the stopping rule

The good-bad assignment decision can be made by computing the percentage of good cases which represents the probability that any random case in the node is a good case. This probability can be treated as a credit score that ranges from zero to one. If the credit score is less than the cut-off score, then all the cases in the node will be assigned into the bad category.

Zhang (2004) uses two steps in tree construction (growing and pruning) to constitute the splitting rule and stopping rule. Suppose that we have observed *p* covariates, denoted by a *p*-vector X, and a response y for n individuals. For the i^{th} individual, the measurements are

$$X_i = \left(X_{i1}, \dots, X_{ip}\right)' \quad and \quad y_i, i = 1, \dots, n. \tag{18}$$

The objective is to model the probability distribution of $P\ (y\ |X)$ or a function of this conditional distribution. The growing step begins with the root node, which is the entire learning sample. The root node is the box on top of the tree in . The most fundamental step in tree growing is to partition the root node into two subgroups, referred to as daughter nodes, such that one daughter node contains mostly bad borrowers (observations with y=0) and the other daughter node mostly good borrowers (observations with y=1). Such a partition is chosen from all possible binary splits based on the profiles of all loan applicants via questions such as: “Is the loan applicant a home owner?” A loan applicant is assigned to the right or left daughter according to whether the answer is yes or no. When all of the loan applicants are assigned to either the left or right daughter nodes, the distribution in terms of the credit score is assessed for both the left and right nodes using typically a node impurity. One such criterion is entropy function

$$it = -pt\log(pt) - (1-pt)\ log(1-pt) \tag{19}$$

where pt is the proportion of bad loan applicants in a specified node *t*. This function is at its lowest level when pt = 0 or 1. 5310131027

other nodes are referred to as internal nodes. More precisely, the quality of a tree, denoted by *T,* is reflected by the quality of its terminal nodes as follows:

$$R(T) = \textstyle\sum_{t\in\tilde{T}} Pr(t)R(t), \tag{20}$$

where $\tilde{T}$ is the set of terminal nodes of tree T and $R(t)$ the within-node misclassification cost of node t.

The ultimate objective of tree pruning is to select a sub-tree of the saturated tree so that the misclassification cost of the selected sub-tree is the lowest on an independent, identically-distributed sample, called a test sample. In practice, we rarely have a test sample. Breiman et al. (1984) proposed using cross validation based on cost-complexity. They defined the number of the terminal nodes of T, denoted by $|\tilde{T}|$, as the complexity of T. A penalizing cost, the so-called complexity parameter, is assigned to a one unit increase in complexity, i.e., one extra terminal node. The sum of all costs becomes the penalty for the tree complexity, and the cost-complexity of a tree is:

$$R_{\propto}(T) = R(T) + \propto |\tilde{T}| \tag{21}$$

where $\propto$ (> 0) is the complexity parameter.

A useful and interesting result from Breiman et al. (1984) is that, for a given complexity parameter, there is a unique smallest sub-tree of the saturated tree that minimizes the cost-complexity measure (3). Furthermore, if $\propto 1 > \propto 2$, the optimally pruned sub-tree corresponding to $\propto 1$ is a sub-tree of the one corresponding to $\propto 2$. Therefore, increasing the complexity parameter produces a finite sequence of nested optimally-pruned sub-trees, which makes the selection of the desirably-sized sub-tree feasible.

Although the introduction of misclassification cost and cost complexity provides a solution to tree pruning, it is usually a subjective and difficult decision to choose the misclassification costs for different errors. Moreover, the final tree can be heavily dependent on such a subjective choice. From a methodological point of view, generalizing the concept of misclassification cost is difficult when we have to deal with more complicated responses. For these reasons, a simpler way for pruning, as described by Segal (1988) and Zhang and Singer (1999), is more preferable. The impurity function can be defined as

$$i_t = -\sum_{j=1}^{J} Pr(y=j)\, log\{Pr(y=j)\} \tag{22}$$

for a J-level y. Everything else in the tree growing step, as described above, is applicable. For tree pruning, the only change to be made is to define the misclassification cost $c(j|k)$ from level k to level j, $j, k = 1, \ldots, J$.

CHAPTER 4

METHODOLOGY

The comparison of the credit scoring methods has been done using the following steps:

4.1 Step 1: Simulate Nine Data Sets of Populations.

4.2 Step 2: Draw 3,000 Data Sets of Samples from Each Population.

4.3 Step 3: Estimate Three Credit Scoring Models Per Each Sample Set.

4.4 Step 4: Test the Model by Using it to Predict the Credit Scores of the Population and Use the Predicted Credit Scores to Compute the K-S, Gini, and Odds Ratio.

4.5 Step 5: Construct a Confusion Matrix at Each Cut-Off Point for Each Model.

4.6 Step 6: Compare the Cross-Validation with the K-S, Gini, and Odds Ratio.

4.7 Step 7: Evaluate the Performance of Different Credit Scoring Models Across Different Methods and Across Samples with Different Characteristics.

The details of each step are presented as follows:

4.1 Step 1: Simulate Nine Data Sets of Populations.

In the first step, nine sets of populations were created; each population set consisted of "good" borrowers and "bad" borrowers. The actual y value for "good" borrowers was "1" and the actual y value for "bad" borrowers was "0." Then a set of information was created. This set of information consisted of ten factors (independent variables) for each borrower ($x_1, x_2, ..., x_{10}$). Each factor was randomly generated following proper distribution with proper value of parameter. For example; if x_1 represented the income of the applicants, it may have followed normal distribution; if x_2 represented residential status (either own or not), it may have followed the

Bernoulli distribution; and if x_3 represented the length of time of employment, it may have followed exponential distribution, etc. (Dryver and Jantra Sukkasem, 2009). These factors represent the personal attributes of the loan applicants and it can be any factors that the bank believed more or less predict whether the loan applicants will default or not.

This research did not specify the factors in the model because the purpose of was not to construct a model but to compare and evaluate the performance of each credit applicant classification model, so that in practice, different banks can choose different sets of factors they believe are important in determining the credit scoring of the applicants. Moreover, it is also flexible for banks to change the sets of influential factors across time.

The results from step 1 are the nine population sets (three different K-S statistics by three different "good" to "bad" ratios). There are three population types based on the parameters used in the simulation so that each population type has a different K-S statistic (high K-S, mid K-S, and low K-S population with K-S statistics equal to 75%, 50%, and 25%, respectively). And for each population type, there are three population sets with different "good" to "bad" ratios:

1) Population 1: 700,000 goods and 300,000 bads (70:30)
2) Population 2: 800,000 goods and 200,000 bads (80:20)
3) Population 3: 900,000 goods and 100,000 bads (90:10)

4.2 Step 2: Draw 3,000 Data Sets of Samples from Each Population

In step 2, various sample sizes (the estimation sample) were drawn from each set of population with 1,000 iterations each.

From each population set, draw 3,000 sets of samples:

1) 1,000 sets of sample 1: 1,000 goods and 1,000 bads
2) 1,000 sets of sample 2: 4,000 goods and 1,000 bads
3) 1,000 sets of sample 3: 9,000 goods and 1,000 bads

Drawing the observations within the same sample set was without replacement. After each sample set was drawn, all of the observations were replaced into the population before drawing the next sample set. For example, Sample 1 was drawn from population set 1, following the proportion of sample type 1 (1,000 goods and 1,000 bads). This step yielded 1,000 sets of sample 1, 1,000 sets of sample 2, and 1,000 sets of sample 3.

Using the same procedure as with population set 1 to draw 3,000 sets of samples from each of the remaining 8 population sets, the research obtained 27,000 sample sets from the nine population sets.

4.3 Step 3: Estimate Three Credit Scoring Models Per Each Sample Set

In step 3, the researcher estimated each credit scoring model by using each sample set as a training set.

The four credit scoring models are

1) LR (Logistic regression)
2) DA (Discriminant analysis)
3) RP1 (Recursive partitioning with restriction on tree growing)
4) RP2 (Recursive partitioning without restriction)

In this step, the research obtained 9 population sets * 3 sample sets per population * 1,000 iterations per sample set * 4 models per iteration. Each of the 27,000 sample sets was used as a training set to estimate each of the 4 models.

In the end, there were 108,000 models formed based on 27,000 sample sets. The numbers of sample sets for each case are summarized in Table 4.1.

Table 4.1 The Numbers of Sample Set to Construct the Four Models

K-S	**Numbers of Sample Set**
High K-S	
Population 1 (70:30)	
Sample 1: 2,000 observations (1,000 goods and 1,000 bads)	1,000
Sample 2: 5,000 observations (4,000 goods and 1,000 bads)	1,000
Sample 3: 10,000 observations (9,000 goods and 1,000 bads)	1,000
Population 2 (80:20)	
Sample 1: 2,000 observations (1,000 goods and 1,000 bads)	1,000
Sample 2: 5,000 observations (4,000 goods and 1,000 bads)	1,000
Sample 3: 10,000 observations (9,000 goods and 1,000 bads)	1,000

Table 4.1 (Continued)

K-S	**Numbers of Sample Set**
Population 3 (90:10)	
Sample 1: 2,000 observations (1,000 goods and 1,000 bads)	1,000
Sample 2: 5,000 observations (4,000 goods and 1,000 bads)	1,000
Sample 3: 10,000 observations (9,000 goods and 1,000 bads)	1,000
Mid K-S	
Population 1 (70:30)	
Sample 1: 2,000 observations (1,000 goods and 1,000 bads)	1,000
Sample 2: 5,000 observations (4,000 goods and 1,000 bads)	1,000
Sample 3: 10,000 observations (9,000 goods and 1,000 bads)	1,000
Population 2 (80:20)	
Sample 1: 2,000 observations (1,000 goods and 1,000 bads)	1,000
Sample 2: 5,000 observations (4,000 goods and 1,000 bads)	1,000
Sample 3: 10,000 observations (9,000 goods and 1,000 bads)	1,000
Population 3 (90:10)	
Sample 1: 2,000 observations (1,000 goods and 1,000 bads)	1,000
Sample 2: 5,000 observations (4,000 goods and 1,000 bads)	1,000
Sample 3: 10,000 observations (9,000 goods and 1,000 bads)	1,000
Low K-S	
Population 1 (70:30)	
Sample 1: 2,000 observations (1,000 goods and 1,000 bads)	1,000
Sample 2: 5,000 observations (4,000 goods and 1,000 bads)	1,000
Sample 3: 10,000 observations (9,000 goods and 1,000 bads)	1,000
Population 2 (80:20)	
Sample 1: 2,000 observations (1,000 goods and 1,000 bads)	1,000
Sample 2: 5,000 observations (4,000 goods and 1,000 bads)	1,000
Sample 3: 10,000 observations (9,000 goods and 1,000 bads)	1,000
Population 3 (90:10)	
Sample 1: 2,000 observations (1,000 goods and 1,000 bads)	1,000
Sample 2: 5,000 observations (4,000 goods and 1,000 bads)	1,000
Sample 3: 10,000 observations (9,000 goods and 1,000 bads)	1,000

4.4 Step 4: Test the Model by Using it to Predict the Credit Scores of the Population and Use the Predicted Credit Scores to Compute the K-S, Gini, and Odds Ratio.

In step 4, the predicted y-values (credit scores) for each sample and its population were computed based on the models estimated from each particular sample set. Then, the K-S, Gini, and odds ratio for each model was computed based on the predicted y-value of each observation. After the K-S, Gini, and odds ratios were computed, it was possible to validate the performance of each model by comparing the K-S, Gini, and odds ratio of each model. For example, to validate the logistic regression model, the performance of the K-S, Gini, and odds ratio of logistic regression models based on different sample sets that were drawn from the same population set was compared with the performance of the other models when used to predict the credit scores of that particular population set.

To compute the K-S, the probability distribution functions (pdf) of the predicted credit scores for the "bad" group and the "good" group were constructed, as illustrated in Figure 4.1. According to Figure 4.1 , the horizontal axis is the predicted credit scores and the vertical axis is the frequency of observing a particular score. The pdf(bad) is right-skewed (the right tail is longer) because most of the bad applicants should get relatively low predicted credit scores, and as a result, the mass of the distribution is concentrated on the left of the figure. On the contrary, the pdf(good) is left-skewed (the left tail is longer) because the majority of the good applicants should get relatively high predicted credit scores, and as a result, the mass of the distribution is concentrated on the right of the figure. Subsequently, the cumulative distribution functions (cdf) of the "bad" group and the "good" group were derived based on their respective pdf.

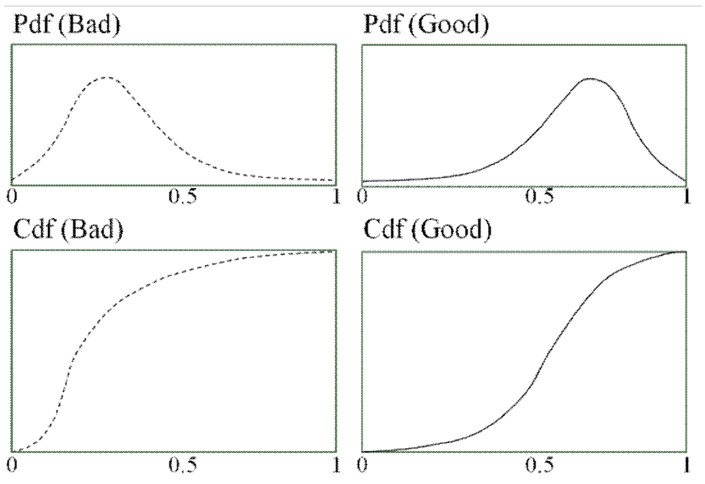

Figure 4.1 Probability Distribution Functions and Cumulative Distribution Functions

Figure 4.2 was formed by combining the cdf(bad) and cdf(good) in Figure 4.1. Consequently, the K-S statistic was calculated from the maximum distance between the two curves. The larger the K-S was, the better the model was able to distinguish between “bad” and “good” borrowers.

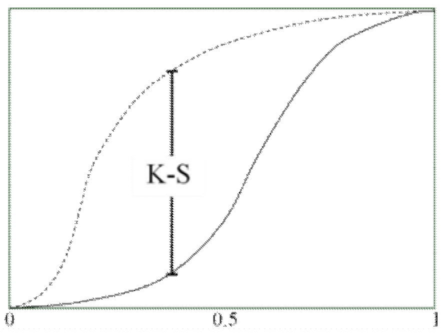

Figure 4.2 Kolmogorov-Smirnov Statistic (K-S)

Finally, the solid curve of the Lorentz diagram was obtained by formulating a scatter plot between the cdf(bad) and the cdf(good), as illustrated in Figure 4.3. The diagonal dotted line represents the points where the cdf(bad) was equal to the cdf(good). If the model had a better performance in terms of distinguishing between the “good” and “bad” borrowers, then the curve was more convex and lay further away from the diagonal line. In this case, the area between the line and the curve will be larger. The Gini coefficient was computed by two times of the area. In conclusion, the higher the Gini coefficient was, the better the model performed.

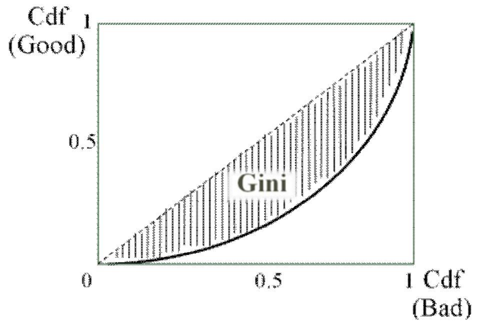

Figure 4.3 Lorentz Diagram and Gini Coefficient

The odds ratio was computed using the number of goods accepted divided by the number of bads accepted.

4.5 Step 5: Construct the Confusion Matrix at Each Cut-Off Point for Each Model

In this step, the performance of each model was compared by constructing the confusion (misclassification) matrix at each cut-off point using the predicted y-value from step 3 and the actual (true) y-value from step 1 and step 2.

The cut-off points were assumed based on what banks use when making a decision on fraud and approving credit cards. For fraud, often banks reject the bottom-scoring 5%. For credit cards, it is between the bottom 10%, 20%, 30%, depending on the bank (Dryver and Jantra Sukkasem, 2009; Dryver, 2011). Thus, assume 4 cut-off points (5%, 10%, 20%, and 30%) to check the percentage of goods and bads rejected at each cut-off point. For example, if we assume a 10% cut-off point, the confusion matrix is formed as shown in Table 4.2. The observation whose predicted credit score is below the cut-off point will be rejected, and the observation whose predicted credit score is above the cut-off point will be accepted.

Table 4.2 Confusion Matrix

Data Set		Predicted	
		Good	**Bad**
Actual	Good	True Positive	False Negative
	Bad	False Positive	True Negative

Where	True Positive = Acceptance of goods
	False Positive = Acceptance of bads
	True Negative = Rejection of bads
	False Negative = Rejection of goods

Based on Table 4.2, it is possible to compute four measures that can be used to gauge the level of misclassification and to compare the performance of different models.

1) Accuracy: (True positives and negatives)/(Total cases)
2) Error rate: (False positives and negatives)/(Total cases)
3) Sensitivity: (True positives)/(Total actual positives)
4) Specificity: (True negatives)/(Total actual negatives)

Based on these measures, a bank can decide, for example, to maximize the rejection of bads (maximize the specificity). In this case, the bank aims to reduce losses. In another case where the bank wishes to get a higher market share and does not mind approving some bads, it can minimize the rejection of goods by choosing the model that maximizes sensitivity (Siddiqi, 2006).

4.6 Step 6: Compare the Cross-Validation with the K-S, Gini, and Odds Ratio

In step 6, the research compares the cross-validation with the K-S, Gini, and odds ratio to assess how informative the K-S, Gini, and odds ratios are. Also, the research investigates the relationships among the K-S, Gini, and odds ratio for the estimation sample and the model performance for validation using the entire population.

4.7 Step 7: Evaluate the Performance of Different Credit Scoring Models Across Different Methods and Across Samples with Different Characteristics.

In the final step, the research evaluates the performance of each model by going back to the results obtained from step 4 – 6. The procedure is illustrated in Figure 4.4. According to Figure 4.4 , after drawing sample sets from each population set, each sample set was used to estimate the different credit scoring models (namely, logistic regression, discriminant analysis, and recursive partitioning). Then the coefficient set from each model was tested on the population data to predict the credit scores. Subsequently, the K-S, Gini, and odds ratios were computed based on the predicted scores. Finally, for each of the three models, the K-S, Gini, odds ratios, and specificity of each model were evaluated across different population “good” to “bad” ratios, sample “good” to “bad” ratios, characteristics of the population, and models. This procedure was repeated for different distribution parameters (high, mid, and low K-S statistics), different “good” to “bad” ratios of populations (population 1, 2, and 3), and different “good” to “bad” ratios of samples (sample 1, 2, and 3).

The R and Java codes for all steps are provided in the appendix. The R codes were written based on Crawley (2007) and the Java codes were written based on Press, Teukolsky, Vetterling, and Flannery (1992).

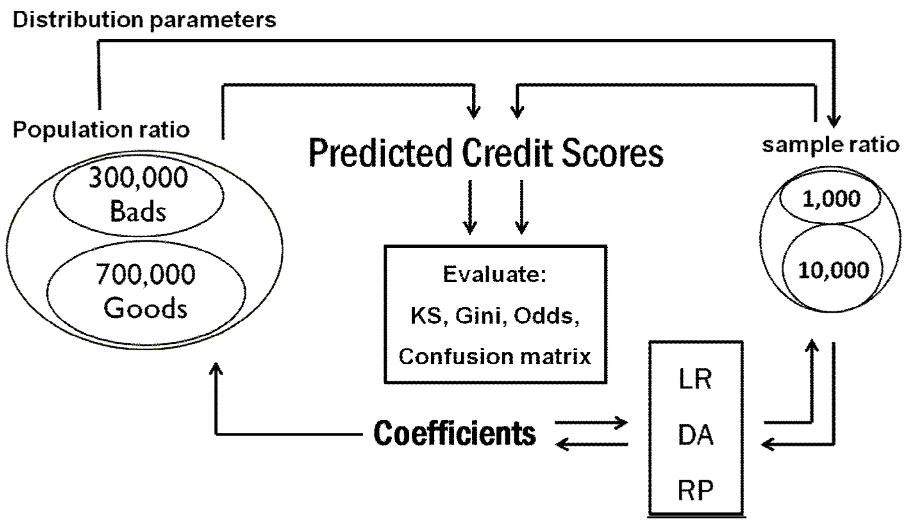

Figure 4.4 Procedure Employed in This Research

CHAPTER 5

ANALYSIS OF RESULTS

This section includes some of the preliminary results as to the degree to which each credit scoring method identifies correctly credit worthy customers. The results are as follows.

The recursive partitioning methods classify the observations into groups and assign the same predicted credit score to the observations that belong in the same group. Logically, the observations with the same predicted credit score should be treated in the same way. So, for the recursive partitioning models, the observations are ranked based on their predicted credit score; then the observations that have a predicted credit score less than or equal to the score at each pre-specified percentage cut-off will be rejected. As a result, the actual percentage cut-offs in many of the iterations are not equal to the pre-specified percentage cut-offs.

Table 5.1 shows the average actual percentage cut-off for each recursive partitioning model. According to this table, RP1 is the recursive partitioning model with the restriction that the tree will stop growing if further growth does not increase the R-squared of the model by more than one percent. RP2 is the recursive partitioning model without such restriction so the tree for RP2 will grow more than the tree for RP1. As a result, RP2 models produce more final nodes and therefore divide the observations into more groups. For this reason, the average actual percentage cut-offs of the RP2 models are much closer to the pre-specified percentage cut-offs compared to those of the RP1 models. A closer look into the average actual percentage cut-offs of the RP2 models reveals that most of them are just about the same as the pre-specified percentage cut-offs. The few exceptions are all of sample type 1, sample type 2 of high KS population type 1 and 2 at 5% cut-off, and some (70H, 70M, 80H, and 80M) of sample type 1 at a 10% cut-off.

In Table 5.1, the two numbers after RP1 or RP2 (70, 80, 90) represent the type of population. 70 is population type one, which has a good:bad ratio equals to 70:30; 80 is population type two, which has a good:bad ratio equals to 80:20; and 90 is population type three, which has a good:bad ratio equals to 90:10. The following letter represents the degree to which the characteristics of "good" set and "bad" set are different from each other. H is high KS, which means they are highly different (KS = 75%), M is mid KS (KS = 50%), and L is low KS (KS = 25%). The last digit represents the type of sample. 1 is sample type 1, 2 is sample type 2, and 3 is sample type 3.

In Table 5.1, some RP1 models running on a population with a low K-S are reported at 100%. The reason for these models to have a 100% percentage cut-off is that when the trees stop growing, there are not enough final nodes to classify the observations based on the pre-determined cut-off percentage. For example, at 10% cut-off, if the tree produces one big final node with members more than 90% of the total observation, when ranking the observations based on the predicted score, using the cut-off score at 10th percentile would result in rejecting all observations. Based on these observations, it is not appropriate to compare the recursive partitioning models to those of the other two models in terms of confusion matrix, accuracy rate, or type I error rate because the actual percentage cut-off rates of the recursive partitioning models are different from those of the other two models.

Table 5.1 Average Actual Cut-Off for the Recursive Partitioning Models

Cut-Off	**5%**	**10%**	**20%**	**30%**	**Cut-Off**	**5%**	**10%**	**20%**	**30%**
RP170H1	18.25%	18.35%	23.48%	32.89%	RP270H1	18.95%	18.95%	20.48%	30.55%
RP170H2	11.03%	12.18%	22.40%	34.65%	RP270H2	10.32%	10.61%	20.31%	30.41%
RP170H3	8.20%	11.79%	22.25%	36.08%	RP270H3	6.77%	10.27%	20.30%	30.94%
RP170M1	17.66%	17.94%	24.02%	37.03%	RP270M1	14.90%	14.90%	20.69%	30.70%
RP170M2	10.16%	12.70%	23.48%	40.22%	RP270M2	5.68%	10.35%	20.66%	30.54%
RP170M3	8.44%	12.77%	25.71%	37.42%	RP270M3	5.24%	10.27%	20.55%	31.35%
RP170L1	5.76%	40.25%	43.81%	47.92%	RP270L1	9.33%	10.72%	20.74%	30.88%
RP170L2	5.57%	46.76%	47.12%	47.28%	RP270L2	5.37%	10.47%	20.71%	30.63%
RP170L3	5.55%	100%	100%	100%	RP270L3	5.47%	10.34%	20.62%	30.94%
RP180H1	13.96%	14.17%	23.12%	32.78%	RP280H1	14.27%	14.27%	20.45%	30.72%
RP180H2	8.04%	11.79%	22.14%	35.22%	RP280H2	7.34%	10.28%	20.34%	30.41%
RP180H3	6.46%	11.50%	24.69%	38.73%	RP280H3	5.19%	10.22%	20.32%	30.18%
RP180M1	14.40%	15.04%	24.87%	36.70%	RP280M1	12.12%	12.14%	20.59%	30.67%
RP180M2	8.27%	12.76%	30.99%	38.31%	RP280M2	5.24%	10.33%	20.46%	30.65%
RP180M3	7.22%	13.69%	35.38%	35.81%	RP280M3	5.20%	10.29%	20.44%	30.31%
RP180L1	34.09%	38.00%	42.04%	48.28%	RP280L1	8.44%	10.59%	20.75%	30.92%
RP180L2	45.47%	45.77%	46.48%	57.60%	RP280L2	5.34%	10.51%	20.61%	30.79%

Table 5.1 (Continued)

Cut-Off	**5%**	**10%**	**20%**	**30%**	**Cut-Off**	**5%**	**10%**	**20%**	**30%**
RP180L3	100%	100%	100%	100%	RP280L3	5.49%	10.33%	20.47%	30.55%
RP190H1	9.60%	11.79%	22.25%	35.27%	RP290H1	9.63%	10.41%	20.51%	30.71%
RP190H2	6.13%	11.60%	25.73%	36.81%	RP290H2	5.16%	10.22%	20.43%	30.31%
RP190H3	5.96%	11.67%	28.38%	35.11%	RP290H3	5.14%	10.18%	20.31%	100%
RP190M1	11.25%	13.43%	26.81%	37.26%	RP290M1	9.19%	10.44%	20.68%	30.94%
RP190M2	6.76%	13.35%	32.62%	34.63%	RP290M2	5.21%	10.33%	20.57%	30.47%
RP190M3	6.69%	12.78%	27.27%	81.64%	RP290M3	5.18%	10.24%	20.46%	69.79%
RP190L1	34.11%	35.98%	40.57%	47.95%	RP290L1	7.62%	10.61%	20.80%	30.94%
RP190L2	56.37%	57.08%	61.45%	67.64%	RP290L2	5.37%	10.49%	20.51%	30.74%
RP190L3	100%	100%	100%	100%	RP290L3	5.26%	10.36%	20.42%	30.47%

Table 5.2 shows the results of cross-validation at the 10% cut-off point on each iteration group from the population sets with high KS, mid KS, and low KS.

The top left and the bottom right cells of each iteration group show the percentage at which the model classifies the observations correctly. According to the confusion matrix, the top left cell and the bottom right cell are also known as the true negative and the true positive, respectively. The other two cells show the percentage at which the model classifies the observations incorrectly. Specifically, the top right cell represents a type I error, where the model suggests that the bank approve the loan to the applicant that actually is a bad borrower (false positive), and the bottom left cell represents a type II error, where the model suggests that the bank reject the good borrower (false negative). The model can be ranked based on one of these four cells or based on the combination of the top left and the bottom right cells (the accuracy) when comparing the performance of each model.

Vertical analysis of the Tables compares the performance of each model across three population types and across three sample types. Vertical analysis across the three population types provides an idea of how different proportions of good borrowers to bad borrowers in a population affect the performance of each model. For example, based on Table 5.2 Panel A, for all types of sample sets, as the ratio of good borrowers to bad borrowers in the population set increases, the percentage of the true positive increases, which means that all of the models can identify "good" borrowers better. However, as the ratio of "good" borrowers to "bad" borrowers in the population set increases, all of the models have worse ability to identify the "bad" borrowers, which is shown by the decrease in the percentage of the true negative.

Vertical analysis across the three different sample types reveals that changing the good:bad ratio of the sample sets does not affect the performance of LR and DA models, of which the average actual cut-offs are approximately equal to the pre-specified cut-offs.

Horizontal analysis of the tables compares the performance of each model within one group of sample at a time. This analysis provides an idea of how each model performs within each population and sample set regime.

The top right cell of each model in Table 5.2 represents a type 1 error, where the model accepts "bad" borrowers. The results show that logistic regression models have the lowest type 1 error for all of the sample types that are drawn from all of the populations with a high KS (from Table 5.2 Panel A). The results are mixed for the populations with a mid KS (from Table 5.2 Panel B). And lastly, the discriminant analysis models have the lowest type 1 error for all of the sample types that are drawn from all the populations with a low KS (from Table 5.2 Panel C).

Similar analysis can be done on Table 5.3. The results from Table 5.3 also shows that logistic regression models perform best (in terms of having the lowest type 1 error) when the populations have a high KS (Table 5.3 Panel A), but the discriminant analysis models perform best when the populations have a low KS. In the mid KS regime, the logistic regression performs best in sample type 1 (when the ratio of good:bad is 1,000:1,000) regardless of the population type; however, the discriminant analysis performs best in sample type 2 (good:bad ratio is 4,000:1,000) and in sample type 3 (good:bad ratio is 9,000:1,000). The ratio of "good" to "bad" in the population does not affect the relative performance of each model.

Table 5.3 shows the results of cross-validation at the 20% cut-off point on each iteration group from the population sets with high KS, mid KS, and low KS, respectively. The interpretation of Table 5.3 is the same as that of Table 5.2. The difference between Table 5.3 and Table 5.2 is the percentage cut-off point.

Table 5.2 Average Performance of Each Model at 10% Cut-Off

3 Types of Sample by KS		**LR Predicted**		**DA Predicted**	
		Y = 0	**Y = 1**	**Y = 0**	**Y = 1**
Panel A: (High KS)					
Sample 1 (1,000 Goods) (pop70:30)					
	Y = 0	9.6932%	20.3068%	9.6913%	20.3087%
Actual	Y = 1	0.3068%	69.6932%	0.3087%	69.6913%
Sample 1 (1,000 Goods) (pop80:20)					
	Y = 0	9.1220%	10.8780%	9.1124%	10.8876%
Actual	Y = 1	0.8780%	79.1220%	0.8876%	79.1124%

Table 5.2 (Continued)

3 Types of Sample by KS		**LR Predicted** Y = 0	Y = 1	**DA Predicted** Y = 0	Y = 1
Panel A: (High KS)					
Sample 1 (1,000 Goods) (pop90:10)					
	Y = 0	6.6052%	3.3948%	6.5818%	3.4182%
Actual	Y = 1	3.3948%	86.6052%	3.4182%	86.5818%
Sample 2 (4,000 Goods) (pop70:30)					
	Y = 0	9.6985%	20.3015%	9.6872%	20.3128%
Actual	Y = 1	0.3015%	69.6985%	0.3128%	69.6872%
Sample 2 (4,000 Goods) (pop80:20)					
	Y = 0	9.1336%	10.8664%	9.0952%	10.9048%
Actual	Y = 1	0.8664%	79.1336%	0.9048%	79.0952%
Sample 2 (4,000 Goods) (pop90:10)					
	Y = 0	6.6225%	3.3775%	6.5375%	3.4625%
Actual	Y = 1	3.3775%	86.6225%	3.4625%	86.5375%
Sample 3 (9,000 Goods) (pop70:30)					
	Y = 0	9.7001%	20.2999%	9.6843%	20.3157%
Actual	Y = 1	0.2999%	69.7001%	0.3157%	69.6843%
Sample 3 (9,000 Goods) (pop80:20)					
	Y = 0	9.1373%	10.8627%	9.0863%	10.9137%
Actual	Y = 1	0.8627%	79.1373%	0.9137%	79.0863%
Sample 3 (9,000 Goods) (pop90:10)					
	Y = 0	6.6278%	3.3722%	6.5182%	3.4818%
Actual	Y = 1	3.3722%	86.6278%	3.4818%	86.5182%

Table 5.2 (Continued)

3 Types of Sample by KS		LR Predicted Y = 0	LR Predicted Y = 1	DA Predicted Y = 0	DA Predicted Y = 1
Panel B: (Mid KS)					
Sample 1 (1,000 Goods) (pop70:30)					
	Y = 0	8.4649%	21.5351%	8.4638%	21.5362%
Actual	Y = 1	1.5351%	68.4649%	1.5362%	68.4638%
Sample 1 (1,000 Goods) (pop80:20)					
	Y = 0	7.2740%	12.7260%	7.1390%	12.8610%
Actual	Y = 1	2.7260%	77.2740%	2.8610%	77.1390%
Sample 1 (1,000 Goods) (pop90:10)					
	Y = 0	4.6857%	5.3143%	4.5224%	5.4776%
Actual	Y = 1	5.3143%	84.6857%	5.4776%	84.5224%
Sample 2 (4,000 Goods) (pop70:30)					
	Y = 0	8.5004%	21.4996%	8.4617%	21.5383%
Actual	Y = 1	1.4996%	68.5004%	1.5383%	68.4617%
Sample 2 (4,000 Goods) (pop80:20)					
	Y = 0	7.2512%	12.7488%	7.2884%	12.7116%
Actual	Y = 1	2.7488%	77.2512%	2.7116%	77.2884%
Sample 2 (4,000 Goods) (pop90:10)					
	Y = 0	4.6353%	5.3647%	4.7075%	5.2925%
Actual	Y = 1	5.3647%	84.6353%	5.2925%	84.7075%
Sample 3 (9,000 Goods) (pop70:30)					
	Y = 0	8.4950%	21.5050%	8.3828%	21.6172%
Actual	Y = 1	1.5050%	68.4950%	1.6172%	68.3828%
Sample 3 (9,000 Goods) (pop80:20)					
	Y = 0	7.2019%	12.7981%	7.2690%	12.7310%
Actual	Y = 1	2.7981%	77.2019%	2.7310%	77.2690%

Table 5.2 (Continued)

3 Types of Sample by KS		LR Predicted Y = 0	LR Predicted Y = 1	DA Predicted Y = 0	DA Predicted Y = 1
Panel B: (Mid KS)					
Sample 3 (9,000 Goods) (pop90:10)					
	Y = 0	4.5851%	5.4149%	4.7409%	5.2591%
Actual	Y = 1	5.4149%	84.5851%	5.2591%	84.7409%
Panel C: (Low KS)					
Sample 1 (1,000 Goods) (pop70:30)					
	Y = 0	5.4589%	24.5411%	5.4598%	24.5402%
Actual	Y = 1	4.5411%	65.4589%	4.5402%	65.4598%
Sample 1 (1,000 Goods) (pop80:20)					
	Y = 0	4.0538%	15.9462%	4.0546%	15.9454%
Actual	Y = 1	5.9462%	74.0538%	5.9454%	74.0546%
Sample 1 (1,000 Goods) (pop90:10)					
	Y = 0	2.2473%	7.7527%	2.2478%	7.7522%
Actual	Y = 1	7.7527%	82.2473%	7.7522%	82.2478%
Sample 2 (4,000 Goods) (pop70:30)					
	Y = 0	5.4850%	24.5150%	5.4890%	24.5110%
Actual	Y = 1	4.5150%	65.4850%	4.5110%	65.4890%
Sample 2 (4,000 Goods) (pop80:20)					
	Y = 0	4.0779%	15.9221%	4.0835%	15.9165%
Actual	Y = 1	5.9221%	74.0779%	5.9165%	74.0835%
Sample 2 (4,000 Goods) (pop90:10)					
	Y = 0	2.2612%	7.7388%	2.2637%	7.7363%
Actual	Y = 1	7.7388%	82.2612%	7.7363%	82.2637%

Table 5.2 (Continued)

3 Types of Sample by KS		LR Predicted Y = 0	LR Predicted Y = 1	DA Predicted Y = 0	DA Predicted Y = 1
Panel C: (Low KS)					
Sample 3 (9,000 Goods) (pop70:30)					
	Y = 0	5.5055%	24.4945%	5.5086%	24.4914%
Actual	Y = 1	4.4945%	65.5055%	4.4914%	65.5086%
Sample 3 (9,000 Goods) (pop80:20)					
	Y = 0	4.0980%	15.9020%	4.1034%	15.8966%
Actual	Y = 1	5.9020%	74.0980%	5.8966%	74.1034%
Sample 3 (9,000 Goods) (pop90:10)					
	Y = 0	2.2698%	7.7302%	2.2724%	7.7276%
Actual	Y = 1	7.7302%	82.2698%	7.7276%	82.2724%

Note: 1. Recursive Partitioning (RP1 and PR2) has been Excluded because their Actual Cut-off Percentages are not Comparable.
2. All Samples have Constant 1,000 bads.

Table 5.3 Average Performance of Each Model at 20% Cut-Off

3 Types of Sample by KS		LR Predicted Y = 0	LR Predicted Y = 1	DA Predicted Y = 0	DA Predicted Y = 1
Panel A: (High KS)					
Sample 1 (1,000 Goods) (pop70:30)					
	Y = 0	18.1073%	11.8927%	18.0800%	11.9200%
Actual	Y = 1	1.8927%	68.1073%	1.9200%	68.0800%
Sample 1 (1,000 Goods) (pop80:20)					
	Y = 0	14.9386%	5.0614%	14.9026%	5.0974%
Actual	Y = 1	5.0614%	74.9386%	5.0974%	74.9026%

Table 5.3 (Continued)

3 Types of Sample by KS		**LR Predicted**		**DA Predicted**	
		Y = 0	**Y = 1**	**Y = 0**	**Y = 1**
Panel A: (High KS)					
Sample 1 (1,000 Goods) (pop90:10)					
	Y = 0	8.5237%	1.4763%	8.5134%	1.4866%
Actual	Y = 1	11.4763%	78.5237%	11.4866%	78.5134%
Sample 2 (4,000 Goods) (pop70:30)					
	Y = 0	18.1298%	11.8702%	18.0286%	11.9714%
Actual	Y = 1	1.8702%	68.1298%	1.9714%	68.0286%
Sample 2 (4,000 Goods) (pop80:20)		Y = 0	Y = 1	Y = 0	Y = 1
	Y = 0	14.9651%	5.0349%	14.8351%	5.1649%
Actual	Y = 1	5.0349%	74.9651%	5.1649%	74.8351%
Sample 2 (4,000 Goods) (pop90:10)					
	Y = 0	8.5355%	1.4645%	8.4821%	1.5179%
Actual	Y = 1	11.4645%	78.5355%	11.5179%	78.4821%
Sample 3 (9,000 Goods) (pop70:30)					
	Y = 0	18.1363%	11.8637%	18.0041%	11.9959%
Actual	Y = 1	1.8637%	68.1363%	1.9959%	68.0041%
Sample 3 (9,000 Goods) (pop80:20)					
	Y = 0	14.9726%	5.0274%	14.8044%	5.1956%
Actual	Y = 1	5.0274%	74.9726%	5.1956%	74.8044%
Sample 3 (9,000 Goods) (pop90:10)					
	Y = 0	8.5384%	1.4616%	8.4653%	1.5347%
Actual	Y = 1	11.4616%	78.5384%	11.5347%	78.4653%

Table 5.3 (Continued)

3 Types of Sample by KS		LR Predicted Y = 0	LR Predicted Y = 1	DA Predicted Y = 0	DA Predicted Y = 1
Panel B: (Mid KS)					
Sample 1 (1,000 Goods) (pop70:30)					
	Y = 0	15.0841%	14.9159%	14.7617%	15.2383%
Actual	Y = 1	4.9159%	65.0841%	5.2383%	64.7617%
Sample 1 (1,000 Goods) (pop80:20)					
	Y = 0	11.6509%	8.3491%	11.3351%	8.6649%
Actual	Y = 1	8.3491%	71.6509%	8.6649%	71.3351%
Sample 1 (1,000 Goods) (pop90:10)					
	Y = 0	6.5926%	3.4074%	6.4001%	3.5999%
Actual	Y = 1	13.4074%	76.5926%	13.5999%	76.4001%
Sample 2 (4,000 Goods) (pop70:30)					
	Y = 0	14.9776%	15.0224%	15.1276%	14.8724%
Actual	Y = 1	5.0224%	64.9776%	4.8724%	65.1276%
Sample 2 (4,000 Goods) (pop80:20)					
	Y = 0	11.5411%	8.4589%	11.6992%	8.3008%
Actual	Y = 1	8.4589%	71.5411%	8.3008%	71.6992%
Sample 2 (4,000 Goods) (pop90:10)					
	Y = 0	6.5227%	3.4773%	6.6213%	3.3787%
Actual	Y = 1	13.4773%	76.5227%	13.3787%	76.6213%
Sample 3 (9,000 Goods) (pop70:30)					
	Y = 0	14.8761%	15.1239%	15.2100%	14.7900%
Actual	Y = 1	5.1239%	64.8761%	4.7900%	65.2100%
Sample 3 (9,000 Goods) (pop80:20)		Y = 0	Y = 1	Y = 0	Y = 1
	Y = 0	11.4407%	8.5593%	11.7898%	8.2102%
Actual	Y = 1	8.5593%	71.4407%	8.2102%	71.7898%

Table 5.3 (Continued)

3 Types of Sample by KS		**LR Predicted Y = 0**	**LR Predicted Y = 1**	**DA Predicted Y = 0**	**DA Predicted Y = 1**
Panel B: (Mid KS)					
Sample 3 (9,000 Goods) (pop90:10)					
	Y = 0	6.4643%	3.5357%	6.6865%	3.3135%
Actual	Y = 1	13.5357%	76.4643%	13.3135%	76.6865%
Panel C: (Low KS)					
Sample 1 (1,000 Goods) (pop70:30)					
	Y = 0	9.7630%	20.2370%	9.7634%	20.2366%
Actual	Y = 1	10.2370%	59.7630%	10.2366%	59.7634%
Sample 1 (1,000 Goods) (pop80:20)					
	Y = 0	7.0218%	12.9782%	7.0224%	12.9776%
Actual	Y = 1	12.9782%	67.0218%	12.9776%	67.0224%
Sample 1 (1,000 Goods) (pop90:10)					
	Y = 0	3.7775%	6.2225%	3.7778%	6.2222%
Actual	Y = 1	16.2225%	73.7775%	16.2222%	73.7778%
Sample 2 (4,000 Goods) (pop70:30)					
	Y = 0	9.7969%	20.2031%	9.7999%	20.2001%
Actual	Y = 1	10.2031%	59.7969%	10.2001%	59.7999%
Sample 2 (4,000 Goods) (pop80:20)					
	Y = 0	7.0532%	12.9468%	7.0573%	12.9427%
Actual	Y = 1	12.9468%	67.0532%	12.9427%	67.0573%
Sample 2 (4,000 Goods) (pop90:10)					
	Y = 0	3.7950%	6.2050%	3.7963%	6.2037%
Actual	Y = 1	16.2050%	73.7950%	16.2037%	73.7963%
Sample 3 (9,000 Goods) (pop70:30)					
	Y = 0	9.8159%	20.1841%	9.8174%	20.1826%
Actual	Y = 1	10.1841%	59.8159%	10.1826%	59.8174%

Table 5.3 (Continued)

3 Types of Sample by KS		LR Predicted Y = 0	LR Predicted Y = 1	DA Predicted Y = 0	DA Predicted Y = 1
Panel C: (Low KS)					
Sample 3 (9,000 Goods) (pop80:20)					
	Y = 0	7.0690%	12.9310%	7.0727%	12.9273%
Actual	Y = 1	12.9310%	67.0690%	12.9273%	67.0727%
Sample 3 (9,000 Goods) (pop90:10)					
	Y = 0	3.8026%	6.1974%	3.8028%	6.1972%
Actual	Y = 1	16.1974%	73.8026%	16.1972%	73.8028%

Note: 1. Recursive Partitioning (RP1 and PR2) has been Excluded because their Actual Cut-off Percentages are not Comparable.
2. All samples have constant 1,000 bads.

Table 5.4 shows the comparison of two models using the cross-validation method at a 10% cut-off point. According to Table 5.4, for populations with high KS, the logistic regression performs best in all population types and all sample types. For a population with mid KS, the logistic regression performs best in all sample types drawn from population type 1 or in sample type 1 drawn from any population type. The discriminant analysis performs best in other sample types. And finally, for a population with a low KS, the discriminant analysis performs best in all population types and sample types.

In Table 5.5, two models are compared using the cross-validation method at a 20% cut-off point. According to Table 5.5, for populations with a high KS, the logistic regression performs best in all population types and all sample types. For population with a mid KS, the logistic regression performs best in samples of type 1 drawn from any population type; however, the discriminant analysis performs best in sample type 2 and 3. And finally, for a population with a low KS, the discriminant analysis performs best in all population types and sample types.

The last column of Table 5.4 and Table 5.5 represents the percentage of number of times that the logistic regression has a higher accuracy rate than the discriminant analysis. For example, in Table 5.4, when using models from sample 1 (1,000 Goods) and testing on population 70:30, high K-S, there are 701 iterations out of 1,000 iterations (70.10% of times) when the logistic regression model obtains a higher accuracy than the discriminant analysis.

Table 5.6 and Table 5.7 depict the type I error (the acceptance of bads) of each model. The last column shows the model that has the lowest error. Comparing Table 5.4 and Table 5.6, the model that performs best based on the accuracy level is also the model that performs best in terms of having the lowest type I error. Comparing Table 5.5 and Table 5.7 yields the same conclusion.

Table 5.4 The Model Performance Based on the Accuracy at 10% Cut-Off

KS	LR	DA	Highest Accuracy	%LR win over DA
High KS				
Population 1 (70:30)				
Sample 1 (1,000 Goods)	79.3863%	79.3825%	LR	70.10%
Sample 2 (4,000 Goods)	79.3969%	79.3744%	LR	99.20%
Sample 3 (9,000 Goods)	79.4002%	79.3685%	LR	100.00%
Population 2 (80:20)				
Sample 1 (1,000 Goods)	88.2441%	88.2248%	LR	82.20%
Sample 2 (4,000 Goods)	88.2672%	88.1904%	LR	99.90%
Sample 3 (9,000 Goods)	88.2746%	88.1727%	LR	100.00%
Population 3 (90:10)				
Sample 1 (1,000 Goods)	93.2104%	93.1635%	LR	89.60%
Sample 2 (4,000 Goods)	93.2450%	93.0750%	LR	99.90%
Sample 3 (9,000 Goods)	93.2556%	93.0363%	LR	100.00%
Mid KS				
Population 1 (70:30)				
Sample 1 (1,000 Goods)	76.9299%	76.9277%	LR	53.40%
Sample 2 (4,000 Goods)	77.0007%	76.9234%	LR	94.80%
Sample 3 (9,000 Goods)	76.9901%	76.7655%	LR	99.50%
Population 2 (80:20)				
Sample 1 (1,000 Goods)	84.5479%	84.2780%	LR	99.70%
Sample 2 (4,000 Goods)	84.5023%	84.5768%	DA	4.70%
Sample 3 (9,000 Goods)	84.4039%	84.5380%	DA	7.50%

Table 5.4 (Continued)

KS	LR	DA	Highest Accuracy	%LR win over DA
Population3 (90:10)				
Sample 1 (1,000 Goods)	89.3713%	89.0447%	LR	100.00%
Sample 2 (4,000 Goods)	89.2706%	89.4149%	DA	0.00%
Sample 3 (9,000 Goods)	89.1701%	89.4819%	DA	0.00%
Low KS				
Population 1 (70:30)				
Sample 1 (1,000 Goods)	70.9178%	70.9196%	DA	36.60%
Sample 2 (4,000 Goods)	70.9699%	70.9780%	DA	23.60%
Sample 3 (9,000 Goods)	71.0110%	71.0172%	DA	29.50%
Population 2 (80:20)				
Sample 1 (1,000 Goods)	78.1077%	78.1092%	DA	38.30%
Sample 2 (4,000 Goods)	78.1558%	78.1669%	DA	15.20%
Sample 3 (9,000 Goods)	78.1960%	78.2068%	DA	20.70%
Population3 (90:10)				
Sample 1 (1,000 Goods)	84.4947%	84.4957%	DA	36.00%
Sample 2 (4,000 Goods)	84.5223%	84.5275%	DA	23.60%
Sample 3 (9,000 Goods)	84.5396%	84.5448%	DA	25.10%

Note: 1. Exclude Recursive Partitioning (RP1 and PR2) because their Actual Cut-off Percentages are not Comparable.

2. All Samples have Constant 1,000 Bads

Table 5.5 The Model Performance Based on the Accuracy at 20% Cut-Off.

KS	LR	DA	Highest Accuracy	%LR win over DA
High KS				
Population 1 (70:30)				
Sample 1 (1,000 Goods)	86.2146%	86.1600%	LR	88.60%
Sample 2 (4,000 Goods)	86.2596%	86.0571%	LR	100.00%
Sample 3 (9,000 Goods)	86.2726%	86.0081%	LR	100.00%
Population 2 (80:20)				
Sample 1 (1,000 Goods)	89.8772%	89.8052%	LR	87.00%
Sample 2 (4,000 Goods)	89.9302%	89.6701%	LR	100.00%
Sample 3 (9,000 Goods)	89.9452%	89.6087%	LR	100.00%
Population 3 (90:10)				
Sample 1 (1,000 Goods)	87.0473%	87.0267%	LR	72.50%
Sample 2 (4,000 Goods)	87.0710%	86.9642%	LR	99.50%
Sample 3 (9,000 Goods)	87.0769%	86.9305%	LR	100.00%
Mid KS				
Population 1 (70:30)				
Sample 1 (1,000 Goods)	80.1682%	79.5234%	LR	100.00%
Sample 2 (4,000 Goods)	79.9551%	80.2553%	DA	0.00%
Sample 3 (9,000 Goods)	79.7522%	80.4200%	DA	0.00%
Population 2 (80:20)				
Sample 1 (1,000 Goods)	83.3017%	82.6703%	LR	100.00%
Sample 2 (4,000 Goods)	83.0822%	83.3984%	DA	0.00%
Sample 3 (9,000 Goods)	82.8814%	83.5796%	DA	0.00%

Table 5.5 (Continued)

KS	LR	DA	Highest Accuracy	%LR win over DA
Mid KS				
Population 3 (90:10)				
Sample 1 (1,000 Goods)	83.1851%	82.8002%	LR	100.00%
Sample 2 (4,000 Goods)	83.0454%	83.2426%	DA	0.00%
Sample 3 (9,000 Goods)	82.9286%	83.3729%	DA	0.00%
Low KS				
Population 1 (70:30)				
Sample 1 (1,000 Goods)	69.5261%	69.5268%	DA	42.70%
Sample 2 (4,000 Goods)	69.5939%	69.5999%	DA	32.60%
Sample 3 (9,000 Goods)	69.6318%	69.6349%	DA	39.50%
Population 2 (80:20)				
Sample 1 (1,000 Goods)	74.0436%	74.0447%	DA	40.60%
Sample 2 (4,000 Goods)	74.1065%	74.1146%	DA	27.60%
Sample 3 (9,000 Goods)	74.1381%	74.1455%	DA	32.20%
Population 3 (90:10)				
Sample 1 (1,000 Goods)	77.5550%	77.5557%	DA	39.90%
Sample 2 (4,000 Goods)	77.5901%	77.5926%	DA	39.90%
Sample 3 (9,000 Goods)	77.6052%	77.6057%	DA	48.40%

Note: 1. Recursive Partitioning (RP1 and PR2) has been Excluded Because their Actual Cut-off Percentages are not Comparable.
2. All samples have constant 1,000 bads.

Table 5.6 The Model Performance Based on Type I Error at 10% Cut-Off

KS	LR	DA	Lowest error
High KS			
Population 1 (70:30)			
Sample 1 (1,000 Goods)	20.3068%	20.3087%	LR
Sample 2 (4,000 Goods)	20.3015%	20.3128%	LR
Sample 3 (9,000 Goods)	20.2999%	20.3157%	LR
Population 2 (80:20)			
Sample 1 (1,000 Goods)	10.8780%	10.8876%	LR
Sample 2 (4,000 Goods)	10.8664%	10.9048%	LR
Sample 3 (9,000 Goods)	10.8627%	10.9137%	LR
Population 3 (90:10)			
Sample 1 (1,000 Goods)	3.3948%	3.4182%	LR
Sample 2 (4,000 Goods)	3.3775%	3.4625%	LR
Sample 3 (9,000 Goods)	3.3722%	3.4818%	LR
Mid KS			
Population 1 (70:30)			
Sample 1 (1,000 Goods)	21.5351%	21.5362%	LR
Sample 2 (4,000 Goods)	21.4996%	21.5383%	LR
Sample 3 (9,000 Goods)	21.5050%	21.6172%	LR
Population 2 (80:20)			
Sample 1 (1,000 Goods)	12.7260%	12.8610%	LR
Sample 2 (4,000 Goods)	12.7488%	12.7116%	DA
Sample 3 (9,000 Goods)	12.7981%	12.7310%	DA
Population 3 (90:10)			
Sample 1 (1,000 Goods)	5.3143%	5.4776%	LR
Sample 2 (4,000 Goods)	5.3647%	5.2925%	DA
Sample 3 (9,000 Goods)	5.4149%	5.2591%	DA
Low KS			
Population 1 (70:30)			
Sample 1 (1,000 Goods)	24.5411%	24.5402%	DA
Sample 2 (4,000 Goods)	24.5150%	24.5110%	DA
Sample 3 (9,000 Goods)	24.4945%	24.4914%	DA
Population 2 (80:20)			
Sample 1 (1,000 Goods)	15.9462%	15.9454%	DA
Sample 2 (4,000 Goods)	15.9221%	15.9165%	DA
Sample 3 (9,000 Goods)	15.9020%	15.8966%	DA

Table 5.6 (Continued)

KS	LR	DA	Lowest error
Population 3 (90:10)			
Sample 1 (1,000 Goods)	7.7527%	7.7522%	DA
Sample 2 (4,000 Goods)	7.7388%	7.7363%	DA
Sample 3 (9,000 Goods)	7.7302%	7.7276%	DA

Note: 1. Recursive Partitioning (RP1 and PR2) has been Excluded Because their Actual Cut-off Percentages are not Comparable.

2. All Samples have Constant 1,000 bads.

Table 5.7 The Model Performance Based on Type I Error at 20% Cut-Off

KS	LR	DA	Lowest error
High KS			
Population 1 (70:30)			
Sample 1 (1,000 Goods)	11.8927%	11.9200%	LR
Sample 2 (4,000 Goods)	11.8702%	11.9714%	LR
Sample 3 (9,000 Goods)	11.8637%	11.9959%	LR
Population 2 (80:20)			
Sample 1 (1,000 Goods)	5.0614%	5.0974%	LR
Sample 2 (4,000 Goods)	5.0349%	5.1649%	LR
Sample 3 (9,000 Goods)	5.0274%	5.1956%	LR
Population 3 (90:10)			
Sample 1 (1,000 Goods)	1.4763%	1.4866%	LR
Sample 2 (4,000 Goods)	1.4645%	1.5179%	LR
Sample 3 (9,000 Goods)	1.4616%	1.5347%	LR
Mid KS			
Population 1 (70:30)			
Sample 1 (1,000 Goods)	14.9159%	15.2383%	LR
Sample 2 (4,000 Goods)	15.0224%	14.8724%	DA
Sample 3 (9,000 Goods)	15.1239%	14.7900%	DA
Population 2 (80:20)			
Sample 1 (1,000 Goods)	8.3491%	8.6649%	LR
Sample 2 (4,000 Goods)	8.4589%	8.3008%	DA
Sample 3 (9,000 Goods)	8.5593%	8.2102%	DA

Table 5.7 (Continued)

KS	LR	DA	Lowest error
Population 3 (90:10)			
Sample 1 (1,000 Goods)	3.4074%	3.5999%	LR
Sample 2 (4,000 Goods)	3.4773%	3.3787%	DA
Sample 3 (9,000 Goods)	3.5357%	3.3135%	DA
Low KS			
Population 1 (70:30)			
Sample 1 (1,000 Goods)	20.2370%	20.2366%	DA
Sample 2 (4,000 Goods)	20.2031%	20.2001%	DA
Sample 3 (9,000 Goods)	20.1841%	20.1826%	DA
Population 2 (80:20)			
Sample 1 (1,000 Goods)	12.9782%	12.9776%	DA
Sample 2 (4,000 Goods)	12.9468%	12.9427%	DA
Sample 3 (9,000 Goods)	12.9310%	12.9273%	DA
Population 3 (90:10)			
Sample 1 (1,000 Goods)	6.2225%	6.2222%	DA
Sample 2 (4,000 Goods)	6.2050%	6.2037%	DA
Sample 3 (9,000 Goods)	6.1974%	6.1972%	DA

Note: 1. Recursive Partitioning (RP1 and PR2) has been Excluded Because their Actual Cut-off Percentages are not Comparable.

2. All Samples have Constant 1,000 bads.

Table 5.8 and Table 5.9 compare four models using Kolmogorov-Smirnov Statistics and Gini coefficients, respectively.

The results from Table 5.8 and Table 5.9 show that, comparing four models under different regimes, the model with the highest KS statistic also has the highest Gini coefficient.

In the high KS population, the logistic regression model has the highest KS statistic and Gini coefficient regardless of the good:bad ratio of the population and the sample.

The mid KS and the low KS populations show opposite results. In the mid KS population, the logistic regression model has the highest KS and Gini in sample type 1 regardless of the population type. But the discriminant analysis model has the highest KS and Gini in sample type 2 and 3.

However, in the low KS population, the discriminant analysis model has the highest KS and Gini in sample type 1, whereas the logistic regression model has the highest KS and Gini in sample type 2 and 3. A change in population type does not alter the results.

Column 7 to column 12 of Table 12 and Table 13 represents the percentage of number of times that one model has a higher K-S than another model. For example, in Table 5.8, when using models from sample 1 (1,000 Goods) and testing on population 70:30 with a high K-S, there are 751 iterations out of 1,000 iterations (75.10% of times) when the logistic regression model obtains a higher K-S than the discriminant analysis, 100% of the time when the logistic regression model obtains a higher K-S than the recursive partitioning models, 100% of the time when the discriminant analysis obtains a higher K-S than the recursive partitioning models, and 23.4% of the time when the restricted recursive partitioning obtains a higher K-S than the unrestricted recursive partitioning.

Based on Table 5.8, in the high K-S population scenario, logistic regression always obtains a higher K-S relative to the discriminant analysis, with a probability of more than 70%, regardless of the population good:bad ratio or the sample good:bad ratio.

In the mid K-S regime, the sample good:bad ratio does affect the results. Logistic regression always obtains a higher K-S relative to the discriminant analysis only in sample 1 (1,000 goods and 1,000 bads), with 100% probability. However, if the sample good:bad ratio varies to be sample 2 (4,000 goods and 1,000 bads) or sample 3 (9,000 goods and 1,000 bads), then the discriminant analysis always obtains a higher K-S, with 100% probability.

The results in the low K-S are opposite from those in the mid K-S regime. In sample 2 (4,000 goods and 1,000 bads) and sample 3 (9,000 goods and 1,000 bads), logistic regression obtains a higher K-S than discriminant 65% to 76% of the total iterations. However, in sample 1 (1,000 goods and 1,000 bads), logistic regression obtains a higher K-S than discriminant only 44% to 46% of the total iterations.

Table 5.8 Kolmogorov-Smirnov Statistics for Each Data Set and Each Model

KS	LR	DA	RP1	RP2	Highest K-S	LR WIN DA	LR WIN RP1	LR WIN RP2	DA WIN RP1	DA WIN RP2	RP1 WIN RP2
High KS											
Population 1 (70:30)											
Sample 1 (1,000 Goods)	72.3987%	72.2803%	62.8430%	63.6740%	LR	75.10%	100.00%	100.00%	100.00%	100.00%	23.40%
Sample 2 (4,000 Goods)	72.5133%	71.9729%	60.6272%	63.6111%	LR	100.00%	100.00%	100.00%	100.00%	100.00%	2.90%
Sample 3 (9,000 Goods)	72.5396%	71.7923%	58.1788%	61.8741%	LR	100.00%	100.00%	100.00%	100.00%	100.00%	2.40%
Population 2 (80:20)											
Sample 1 (1,000 Goods)	72.5801%	72.4397%	62.7316%	63.7673%	LR	78.10%	100.00%	100.00%	100.00%	100.00%	95.50%
Sample 2 (4,000 Goods)	72.7057%	72.1062%	60.5565%	63.6844%	LR	99.60%	100.00%	100.00%	100.00%	100.00%	96.90%
Sample 3 (9,000 Goods)	72.7394%	71.9381%	57.9825%	62.0103%	LR	100.00%	100.00%	100.00%	100.00%	100.00%	99.20%
Population 3 (90:10)											
Sample 1 (1,000 Goods)	72.5273%	72.4215%	62.9245%	63.9401%	LR	73.30%	100.00%	100.00%	100.00%	100.00%	94.00%
Sample 2 (4,000 Goods)	72.6524%	72.0955%	60.8177%	63.8571%	LR	99.40%	100.00%	100.00%	100.00%	100.00%	99.50%
Sample 3 (9,000 Goods)	72.6837%	71.9150%	58.2382%	62.2018%	LR	100.00%	100.00%	100.00%	100.00%	100.00%	29.90%
Mid KS											
Population 1 (70:30)											
Sample 1 (1,000 Goods)	52.3734%	50.5660%	46.1017%	42.8599%	LR	100.00%	100.00%	100.00%	100.00%	100.00%	19.90%
Sample 2 (4,000 Goods)	51.7477%	52.6577%	45.1975%	41.7045%	DA	0.00%	100.00%	100.00%	100.00%	100.00%	2.10%
Sample 3 (9,000 Goods)	51.1789%	53.2783%	44.4041%	39.4519%	DA	0.00%	100.00%	100.00%	100.00%	100.00%	1.50%
Population 2 (80:20)											
Sample 1 (1,000 Goods)	52.2946%	50.5532%	45.9664%	42.7891%	LR	100.00%	100.00%	100.00%	100.00%	100.00%	95.50%
Sample 2 (4,000 Goods)	51.7134%	52.5819%	45.0159%	41.5933%	DA	0.00%	100.00%	100.00%	100.00%	100.00%	96.90%
Sample 3 (9,000 Goods)	51.1658%	53.2110%	44.1232%	39.3684%	DA	0.00%	100.00%	100.00%	100.00%	100.00%	99.10%

Table 5.8 (Continued)

KS	LR	DA	RP1	RP2	Highest K-S	LR WIN DA	LR WIN RP1	LR WIN RP2	DA WIN RP1	DA WIN RP2	RP1 WIN RP2
Mid KS											
Population 3 (90:10)											
Sample 1 (1,000 Goods)	52.4615%	50.6524%	46.1998%	43.0115%	LR	100.00%	100.00%	100.00%	100.00%	100.00%	94.40%
Sample 2 (4,000 Goods)	51.8495%	52.7508%	45.1455%	41.8454%	DA	0.00%	100.00%	100.00%	100.00%	100.00%	96.20%
Sample 3 (9,000 Goods)	51.2999%	53.3257%	44.6899%	39.7526%	DA	0.00%	100.00%	100.00%	100.00%	100.00%	22.50%
Low KS											
Population 1 (70:30)											
Sample 1 (1,000 Goods)	23.2939%	23.2967%	16.9416%	14.0439%	DA	44.10%	100.00%	100.00%	100.00%	100.00%	23.40%
Sample 2 (4,000 Goods)	23.4752%	23.4523%	16.0681%	13.4905%	LR	66.90%	100.00%	100.00%	100.00%	100.00%	3.00%
Sample 3 (9,000 Goods)	23.4845%	23.4464%	11.7243%	12.0834%	LR	72.60%	100.00%	100.00%	100.00%	100.00%	1.50%
Population 2 (80:20)											
Sample 1 (1,000 Goods)	23.3639%	23.3649%	17.1838%	14.3138%	DA	46.60%	100.00%	100.00%	100.00%	100.00%	95.90%
Sample 2 (4,000 Goods)	23.5654%	23.5341%	16.0695%	13.6851%	LR	71.90%	100.00%	100.00%	100.00%	100.00%	98.00%
Sample 3 (9,000 Goods)	23.5675%	23.5171%	11.9050%	12.4318%	LR	75.90%	100.00%	100.00%	100.00%	100.00%	99.10%
Population 3 (90:10)											
Sample 1 (1,000 Goods)	23.5133%	23.5134%	17.3244%	14.3629%	DA	48.40%	100.00%	100.00%	100.00%	100.00%	93.40%
Sample 2 (4,000 Goods)	23.6704%	23.6484%	14.9799%	13.8151%	LR	65.70%	100.00%	100.00%	100.00%	100.00%	72.10%
Sample 3 (9,000 Goods)	23.7046%	23.6656%	11.8352%	12.6605%	LR	71.70%	100.00%	100.00%	100.00%	100.00%	11.90%

Note: All Samples have Constant 1,000 bads.

Table 5.9 Gini Coefficients for Each Data Set and Each Model

KS	LR	DA	RP1	RP2	Highest Gini	LR WIN DA	LR WIN RP1	LR WIN RP2	DA WIN RP1	DA WIN RP2	RP1 WIN RP2
High KS											
Population 1 (70:30)											
Sample 1 (1,000 Goods)	87.2937%	87.2328%	74.4482%	76.1894%	LR	71.60%	100.00%	100.00%	100.00%	100.00%	7.30%
Sample 2 (4,000 Goods)	87.3715%	87.0702%	72.2831%	74.1779%	LR	98.30%	100.00%	100.00%	100.00%	100.00%	11.30%
Sample 3 (9,000 Goods)	87.3833%	86.9589%	70.3399%	69.5488%	LR	99.90%	100.00%	100.00%	100.00%	100.00%	65.80%
Population 2 (80:20)											
Sample 1 (1,000 Goods)	87.3898%	87.3197%	74.4873%	76.2870%	LR	73.70%	100.00%	100.00%	100.00%	100.00%	88.30%
Sample 2 (4,000 Goods)	87.4623%	87.1468%	72.2323%	74.2565%	LR	98.80%	100.00%	100.00%	100.00%	100.00%	77.30%
Sample 3 (9,000 Goods)	87.4807%	87.0473%	70.4567%	70.0991%	LR	99.80%	100.00%	100.00%	100.00%	100.00%	99.40%
Population 3 (90:10)											
Sample 1 (1,000 Goods)	87.4189%	87.3603%	74.5482%	76.4176%	LR	71.10%	100.00%	100.00%	100.00%	100.00%	98.20%
Sample 2 (4,000 Goods)	87.4978%	87.1930%	72.3784%	74.4513%	LR	98.30%	100.00%	100.00%	100.00%	100.00%	99.60%
Sample 3 (9,000 Goods)	87.5165%	87.0910%	70.5213%	69.9040%	LR	100.00%	100.00%	100.00%	100.00%	100.00%	0.10%
Mid KS											
Population 1 (70:30)											
Sample 1 (1,000 Goods)	68.0491%	66.4155%	56.2124%	54.2028%	LR	100.00%	100.00%	100.00%	100.00%	100.00%	7.00%
Sample 2 (4,000 Goods)	67.5423%	68.3015%	53.2435%	51.7001%	DA	0.00%	100.00%	100.00%	100.00%	100.00%	7.90%
Sample 3 (9,000 Goods)	67.0286%	68.7361%	51.4390%	46.2540%	DA	0.00%	100.00%	100.00%	100.00%	100.00%	58.40%
Population 2 (80:20)											
Sample 1 (1,000 Goods)	67.9479%	66.3292%	56.2622%	54.0921%	LR	100.00%	100.00%	100.00%	100.00%	100.00%	90.90%
Sample 2 (4,000 Goods)	67.4527%	68.2010%	53.7675%	51.5161%	DA	0.00%	100.00%	100.00%	100.00%	100.00%	90.70%
Sample 3 (9,000 Goods)	66.9339%	68.6187%	52.3335%	46.1942%	DA	0.00%	100.00%	100.00%	100.00%	100.00%	100.00%

Table 5.9 (Continued)

KS	LR	DA	RP1	RP2	Highest Gini	LR WIN DA	LR WIN RP1	LR WIN RP2	DA WIN RP1	DA WIN RP2	RP1 WIN RP2
Mid KS											
Population 3 (90:10)											
Sample 1 (1,000 Goods)	68.1108%	66.4534%	56.3188%	54.3443%	LR	100.00%	100.00%	100.00%	100.00%	100.00%	98.50%
Sample 2 (4,000 Goods)	67.5932%	68.3563%	54.2205%	51.8294%	DA	0.00%	100.00%	100.00%	100.00%	100.00%	98.00%
Sample 3 (9,000 Goods)	67.0815%	68.7771%	51.2208%	46.6166%	DA	0.00%	100.00%	100.00%	100.00%	100.00%	0.00%
Low KS											
Population 1 (70:30)											
Sample 1 (1,000 Goods)	32.5636%	32.5680%	22.6521%	18.6774%	DA	43.60%	100.00%	100.00%	100.00%	100.00%	6.90%
Sample 2 (4,000 Goods)	32.8199%	32.7844%	20.7682%	17.5938%	LR	71.70%	100.00%	100.00%	100.00%	100.00%	7.90%
Sample 3 (9,000 Goods)	32.8692%	32.8053%	11.7243%	14.4598%	LR	77.10%	100.00%	100.00%	100.00%	100.00%	61.40%
Population 2 (80:20)											
Sample 1 (1,000 Goods)	32.6888%	32.6928%	22.9470%	19.0403%	DA	43.80%	100.00%	100.00%	100.00%	100.00%	89.30%
Sample 2 (4,000 Goods)	32.9684%	32.9288%	21.0094%	17.8651%	LR	72.70%	100.00%	100.00%	100.00%	100.00%	93.60%
Sample 3 (9,000 Goods)	33.0090%	32.9407%	11.9050%	14.9179%	LR	79.00%	100.00%	100.00%	100.00%	100.00%	98.70%
Population 3 (90:10)											
Sample 1 (1,000 Goods)	32.7931%	32.7975%	23.1164%	19.0913%	DA	41.30%	100.00%	100.00%	100.00%	100.00%	98.80%
Sample 2 (4,000 Goods)	33.0194%	32.9829%	18.7647%	18.0132%	LR	71.30%	100.00%	100.00%	100.00%	100.00%	71.60%
Sample 3 (9,000 Goods)	33.0822%	33.0191%	11.8352%	15.1823%	LR	77.80%	100.00%	100.00%	100.00%	100.00%	0.00%

Note: All Samples have Constant 1,000 bads.

Table 5.10 to Table 5.13 show the odds at a 5%, 10%, 20%, and 30% cut-off point respectively. Odds is the proportion with the numerator as the acceptance of good loans (true positive) and the denominator as the acceptance of bad loans (false positive). The numerator represents the power of the model and the denominator represents the type I error of the model. Therefore, this ratio standardizes the error of the model. The higher the ratio is, the better the performance of the model is.

It can be observed from Table 5.10 that in the high K-S population, the logistic regression models have a relatively higher odds ratio when compared to other three models except for one case (sample type 1 drawn from population type 1); in the mid K-S population, the result is mixed; and in the low K-S population, discriminant analysis models have a relatively higher odds ratio in all cases.

The results of the 10% cut-off in Table 5.11 and 20% cut-off in Table produce a clear pattern for all types of populations. In the high K-S population, the logistic regression models perform best in all cases, whereas in the low K-S populations, the discriminant analysis models perform best in all cases. In the mid K-S population, the logistic regression models perform best for either sample type 1 drawn from any population type, or any sample type drawn from population type 1. The discriminant analysis models perform best in the remaining cases.

Finally, Table 5.13, which shows the relative performance of each model at 30% cut-off, produces the same results for the high and mid K-S population cases compared to the results from Table 5.11 and Table 5.12. Logistic regression models perform best in all cases under a high K-S population. Under a mid K-S population, logistic regression models perform best if the sample group is either sample type 1 or population type 1. However, under a low K-S population, the results from Table are different from those of Table 5.11 and Table 5.12. When the population has a low K-S, discriminant analysis models perform best for sample type 1 drawn from any population type; however, logistic analysis models perform best for sample type 3 drawn from any population type. The results are mixed in sample type 2.

Table 5.10 Odds Ratio of Each Data Set and Each Model for Cut-Off at 5%

KS	DA	LR	RP1	RP2	Highest Odds
High KS					
Population 1 (70:30)					
Sample 1 (1,000 Goods)*	2.7899	2.7898	4.3926	4.6251	DA
Sample 2 (4,000 Goods)*	2.7899	2.7901	3.4435	3.3664	LR
Sample 3 (9,000 Goods)*	2.7899	2.7901	3.1100	2.9570	LR
Population 2 (80:20)					
Sample 1 (1,000 Goods)*	5.2607	5.2608	7.5897	7.9552	LR
Sample 2 (4,000 Goods)*	5.2597	5.2618	5.9509	5.7894	LR
Sample 3 (9,000 Goods)	5.2589	5.2621	5.5255	5.2034	LR
Population 3 (90:10)					
Sample 1 (1,000 Goods)*	15.4016	15.4199	17.1624	17.9832	LR
Sample 2 (4,000 Goods)	15.3631	15.4466	14.5787	13.9425	LR
Sample 3 (9,000 Goods)	15.3445	15.4553	14.7763	14.2569	LR
Mid KS					
Population 1 (70:30)					
Sample 1 (1,000 Goods)*	2.7211	2.7029	3.7234	3.3784	DA
Sample 2 (4,000 Goods)	2.7008	2.7134	3.1093	2.7454	LR
Sample 3 (9,000 Goods)	2.6892	2.7192	2.9855	2.7224	LR
Population 2 (80:20)					
Sample 1 (1,000 Goods)*	4.9447	4.9129	6.3950	5.7913	DA
Sample 2 (4,000 Goods)	4.9079	4.9374	5.4155	4.8716	LR
Sample 3 (9,000 Goods)	4.8732	4.9458	5.2659	4.8952	LR
Population 3 (90:10)					
Sample 1 (1,000 Goods)*	12.3741	12.4210	14.5247	13.0585	LR
Sample 2 (4,000 Goods)	12.4242	12.4495	12.6819	11.8318	LR
Sample 3 (9,000 Goods)	12.3048	12.4247	12.7825	11.9106	LR
Low KS					
Population 1 (70:30)					
Sample 1 (1,000 Goods)*	2.5165	2.5164	2.6529	2.5710	DA
Sample 2 (4,000 Goods)	2.5191	2.5187	2.6515	2.5120	DA
Sample 3 (9,000 Goods)	2.5212	2.5208	2.6506	2.5030	DA
Population 2 (80:20)					
Sample 1 (1,000 Goods)*	4.3634	4.3631	5.3972	4.4194	DA
Sample 2 (4,000 Goods)	4.3693	4.3682	5.6629	4.3376	DA
Sample 3 (9,000 Goods)	4.3738	4.3727	NA	4.3287	DA
Population 3 (90:10)					
Sample 1 (1,000 Goods)*	9.9380	9.9373	12.2691	9.9406	DA
Sample 2 (4,000 Goods)	9.9561	9.9530	NA	9.8255	DA
Sample 3 (9,000 Goods)	9.9659	9.9631	NA	9.7779	DA

Note: 1. Recursive Partitioning (RP1 and RP2) has been Excluded Due to Difference in Percentage Cut-off.
2. All Samples have Constant 1,000 bads.

Table 5.11 Odds Ratio of Each Data Set and Each Model for Cut-Off at 10%

KS	DA	LR	RP1	RP2	Highest Odds
High KS					
Population 1 (70:30)					
Sample 1 (1,000 Goods)	3.4316	3.4320	4.4129	4.6251	LR
Sample 2 (4,000 Goods)	3.4307	3.4332	3.5848	3.4033	LR
Sample 3 (9,000 Goods)	3.4301	3.4335	3.5638	3.3836	LR
Population 2 (80:20)					
Sample 1 (1,000 Goods)	7.2663	7.2736	7.6535	7.9552	LR
Sample 2 (4,000 Goods)	7.2533	7.2824	7.1562	6.7743	LR
Sample 3 (9,000 Goods)	7.2465	7.2852	7.1899	6.8678	LR
Population 3 (90:10)					
Sample 1 (1,000 Goods)	25.3311	25.5118	19.1854	18.7842	LR
Sample 2 (4,000 Goods)	24.9943	25.6473	21.0339	20.3455	LR
Sample 3 (9,000 Goods)	24.8497	25.6891	21.1507	20.4947	LR
Mid KS					
Population 1 (70:30)					
Sample 1 (1,000 Goods)	3.1790	3.1792	3.7472	3.3784	LR
Sample 2 (4,000 Goods)	3.1786	3.1861	3.3260	3.1051	LR
Sample 3 (9,000 Goods)	3.1633	3.1851	3.3578	3.1099	LR
Population 2 (80:20)					
Sample 1 (1,000 Goods)	5.9980	6.0721	6.5045	5.7941	LR
Sample 2 (4,000 Goods)	6.0802	6.0595	6.2922	5.7105	DA
Sample 3 (9,000 Goods)	6.0694	6.0323	6.5433	5.7008	DA
Population 3 (90:10)					
Sample 1 (1,000 Goods)	15.4315	15.9360	15.5737	13.4864	LR
Sample 2 (4,000 Goods)	16.0053	15.7769	16.3935	14.0468	DA
Sample 3 (9,000 Goods)	16.1134	15.6211	16.1816	13.8742	DA
Low KS					
Population 1 (70:30)					
Sample 1 (1,000 Goods)	2.6675	2.6673	3.2085	2.5905	DA
Sample 2 (4,000 Goods)	2.6718	2.6712	3.2737	2.5982	DA
Sample 3 (9,000 Goods)	2.6748	2.6743	NA	2.5719	DA
Population 2 (80:20)					
Sample 1 (1,000 Goods)	4.6443	4.6440	5.5136	4.4718	DA
Sample 2 (4,000 Goods)	4.6545	4.6526	5.6709	4.4910	DA
Sample 3 (9,000 Goods)	4.6616	4.6597	NA	4.4475	DA
Population 3 (90:10)					
Sample 1 (1,000 Goods)	10.6098	10.6090	12.4042	10.1089	DA
Sample 2 (4,000 Goods)	10.6336	10.6297	NA	10.1601	DA
Sample 3 (9,000 Goods)	10.6466	10.6427	NA	10.0652	DA

Note: 1. Recursive Partitioning (RP1 and PR2) has been Excluded Because their Actual Cut-off Percentages are not Comparable.
2. All Samples have Constant 1,000 bads.

Table 5.12 Odds Ratio of Each Data Set and Each Model for Cut-Off at 20%

KS	DA	LR	RP1	RP2	Highest Odds
High KS					
Population 1 (70:30)					
Sample 1 (1,000 Goods)	5.7115	5.7268	5.3534	4.9069	LR
Sample 2 (4,000 Goods)	5.6826	5.7396	5.4726	5.1432	LR
Sample 3 (9,000 Goods)	5.6690	5.7433	5.4453	5.1767	LR
Population 2 (80:20)					
Sample 1 (1,000 Goods)	14.6954	14.8063	11.8015	10.7797	LR
Sample 2 (4,000 Goods)	14.4903	14.8892	11.8255	11.4815	LR
Sample 3 (9,000 Goods)	14.3985	14.9129	11.9785	11.2216	LR
Population 3 (90:10)					
Sample 1 (1,000 Goods)	52.8220	53.1914	34.6144	33.0982	LR
Sample 2 (4,000 Goods)	51.7161	53.6265	35.2769	33.8799	LR
Sample 3 (9,000 Goods)	51.1366	53.7366	35.0524	31.3770	LR
Mid KS					
Population 1 (70:30)					
Sample 1 (1,000 Goods)	4.2501	4.3635	4.3458	3.7992	LR
Sample 2 (4,000 Goods)	4.3791	4.3255	4.4532	3.9252	DA
Sample 3 (9,000 Goods)	4.4091	4.2897	4.6063	3.8386	DA
Population 2 (80:20)					
Sample 1 (1,000 Goods)	8.2335	8.5826	8.3668	7.0125	LR
Sample 2 (4,000 Goods)	8.6379	8.4580	9.6147	7.1733	DA
Sample 3 (9,000 Goods)	8.7440	8.3470	10.0897	6.9403	DA
Population 3 (90:10)					
Sample 1 (1,000 Goods)	21.2264	22.4824	21.9376	17.3149	LR
Sample 2 (4,000 Goods)	22.6793	22.0091	24.0429	17.4797	DA
Sample 3 (9,000 Goods)	23.1439	21.6282	21.5374	16.7365	DA
Low KS					
Population 1 (70:30)					
Sample 1 (1,000 Goods)	2.9533	2.9532	3.2688	2.7118	DA
Sample 2 (4,000 Goods)	2.9604	2.9598	3.2789	2.7211	DA
Sample 3 (9,000 Goods)	2.9638	2.9635	NA	2.6760	DA
Population 2 (80:20)					
Sample 1 (1,000 Goods)	5.1645	5.1643	5.6305	4.6912	DA
Sample 2 (4,000 Goods)	5.1811	5.1792	5.6903	4.6982	DA
Sample 3 (9,000 Goods)	5.1885	5.1867	NA	4.6243	DA
Population 3 (90:10)					
Sample 1 (1,000 Goods)	11.8575	11.8568	12.7008	10.6062	DA
Sample 2 (4,000 Goods)	11.8956	11.8930	NA	10.6317	DA
Sample 3 (9,000 Goods)	11.9092	11.9087	NA	10.4702	DA

Note: 1. Recursive Partitioning (RP1 and PR2) has been Excluded Because their Actual Cut-off Percentages are not Comparable.
2. All Samples have Constant 1,000 bads.

Table 5.13 Odds Ratio of Each Data Set and Each Model for Cut-Off at 30%

KS	DA	LR	RP1	RP2	Highest Odds
High KS					
Population 1 (70:30)					
Sample 1 (1,000 Goods)	10.3915	10.4674	8.3365	7.6628	LR
Sample 2 (4,000 Goods)	10.2357	10.5275	8.3714	7.9318	LR
Sample 3 (9,000 Goods)	10.1596	10.5438	8.1985	7.6822	LR
Population 2 (80:20)					
Sample 1 (1,000 Goods)	27.6742	27.8773	18.1650	17.4030	LR
Sample 2 (4,000 Goods)	27.1310	28.0721	17.0274	17.1616	LR
Sample 3 (9,000 Goods)	26.8625	28.1164	17.7956	16.0216	LR
Population 3 (90:10)					
Sample 1 (1,000 Goods)	90.0091	90.2243	53.2787	50.4986	LR
Sample 2 (4,000 Goods)	88.7174	90.8999	47.3906	46.9354	LR
Sample 3 (9,000 Goods)	87.8906	91.0450	43.0406	NA	LR
Mid KS					
Population 1 (70:30)					
Sample 1 (1,000 Goods)	5.5402	5.8055	5.8096	4.6606	LR
Sample 2 (4,000 Goods)	5.8486	5.7110	6.0808	4.6571	DA
Sample 3 (9,000 Goods)	5.9404	5.6279	5.6742	4.4592	DA
Population 2 (80:20)					
Sample 1 (1,000 Goods)	10.7680	11.3825	10.7863	8.5983	LR
Sample 2 (4,000 Goods)	11.4886	11.1673	10.8136	8.4303	DA
Sample 3 (9,000 Goods)	11.7161	10.9740	10.1479	7.9669	DA
Population 3 (90:10)					
Sample 1 (1,000 Goods)	27.5745	29.4682	26.2854	21.0197	LR
Sample 2 (4,000 Goods)	29.7699	28.8132	24.7275	20.2392	DA
Sample 3 (9,000 Goods)	30.4238	28.2390	NA	NA	DA
Low KS					
Population 1 (70:30)					
Sample 1 (1,000 Goods)	3.2460	3.2460	3.3327	2.8274	DA
Sample 2 (4,000 Goods)	3.2561	3.2558	3.2809	2.8244	DA
Sample 3 (9,000 Goods)	3.2587	3.2588	NA	2.7635	LR
Population 2 (80:20)					
Sample 1 (1,000 Goods)	5.6889	5.6887	5.8079	4.8961	DA
Sample 2 (4,000 Goods)	5.7092	5.7102	5.8027	4.8758	LR
Sample 3 (9,000 Goods)	5.7132	5.7157	NA	4.7772	LR
Population 3 (90:10)					
Sample 1 (1,000 Goods)	13.1137	13.1130	13.1787	11.0756	DA
Sample 2 (4,000 Goods)	13.1581	13.1600	NA	11.0369	LR
Sample 3 (9,000 Goods)	13.1681	13.1732	NA	10.8289	LR

Note: 1. Recursive Partitioning (RP1 and PR2) has been Excluded Because their Actual Cut-off Percentages are not Comparable.
2. All Samples have Constant 1,000 bads.

Table 5.14 The Optimal Model Based on Each Criterion in Each Scenario

K-S	Odds5%	Acc10%	Err10%	Odds10%	Acc20%	Err20%	Odds20%	Odds30%	KS	Gini
High K-S										
Population 1 (70:30)										
Sample 1 (1,000 Goods)	DA	LR	LR	LR	LR	LR	LR	LR	LR	LR
Sample 2 (4,000 Goods)	LR	LR	LR	LR	LR	LR	LR	LR	LR	LR
Sample 3 (9,000 Goods)	LR	LR	LR	LR	LR	LR	LR	LR	LR	LR
Population 2 (80:20)										
Sample 1 (1,000 Goods)	LR	LR	LR	LR	LR	LR	LR	LR	LR	LR
Sample 2 (4,000 Goods)	LR	LR	LR	LR	LR	LR	LR	LR	LR	LR
Sample 3 (9,000 Goods)	LR	LR	LR	LR	LR	LR	LR	LR	LR	LR
Population 3 (90:10)										
Sample 1 (1,000 Goods)	LR	LR	LR	LR	LR	LR	LR	LR	LR	LR
Sample 2 (4,000 Goods)	LR	LR	LR	LR	LR	LR	LR	LR	LR	LR
Sample 3 (9,000 Goods)	LR	LR	LR	LR	LR	LR	LR	LR	LR	LR
Mid KS										
Population 1 (70:30)										
Sample 1 (1,000 Goods)	DA	LR	LR	LR	LR	LR	LR	LR	LR	LR
Sample 2 (4,000 Goods)	LR	LR	LR	LR	DA	DA	DA	DA	DA	DA
Sample 3 (9,000 Goods)	LR	LR	LR	LR	DA	DA	DA	DA	DA	DA
Population 2 (80:20)										
Sample 1 (1,000 Goods)	DA	LR	LR	LR	LR	LR	LR	LR	LR	LR
Sample 2 (4,000 Goods)	LR	DA	DA	DA	DA	DA	DA	DA	DA	DA
Sample 3 (9,000 Goods)	LR	DA	DA	DA	DA	DA	DA	DA	DA	DA
Population 3 (90:10)										
Sample 1 (1,000 Goods)	LR	LR	LR	LR	LR	LR	LR	LR	LR	LR
Sample 2 (4,000 Goods)	LR	DA	DA	DA	DA	DA	DA	DA	DA	DA
Sample 3 (9,000 Goods)	LR	DA	DA	DA	DA	DA	DA	DA	DA	DA

Table 5.14 (Continued)

K-S	Odds5%	Acc10%	Err10%	Odds10%	Acc20%	Err20%	Odds20%	Odds30%	KS	Gini
Low K-S										
Population 1 (70:30)										
Sample 1 (1,000 Goods)	DA	DA	DA	DA	DA	DA	DA	DA	DA	DA
Sample 2 (4,000 Goods)	DA	DA	DA	DA	DA	DA	DA	DA	LR	LR
Sample 3 (9,000 Goods)	DA	DA	DA	DA	DA	DA	DA	LR	LR	LR
Population 2 (80:20)										
Sample 1 (1,000 Goods)	DA	DA	DA	DA	DA	DA	DA	DA	DA	DA
Sample 2 (4,000 Goods)	DA	DA	DA	DA	DA	DA	DA	LR	LR	LR
Sample 3 (9,000 Goods)	DA	DA	DA	DA	DA	DA	DA	LR	LR	LR
Population 3 (90:10)										
Sample 1 (1,000 Goods)	DA	DA	DA	DA	DA	DA	DA	DA	DA	DA
Sample 2 (4,000 Goods)	DA	DA	DA	DA	DA	DA	DA	LR	LR	LR
Sample 3 (9,000 Goods)	DA	DA	DA	DA	DA	DA	DA	LR	LR	LR

Note: 1. Recursive Partitioning (RP1 and RP2) has been Excluded in Odds, Accuracy rate, and Error Rate Due to Difference in Actual Percentage Cut-off.

2. All samples have constant 1,000 bads.

CHAPTER 6

CONCLUDING REMARKS

This chapter includes the underlying assumptions of this study, the discussion, followed by the findings and suggestions regarding research ideas for the future.

The main assumption of this research was that modeling technique performance is not data specific and that the key to understanding modeling technique performance can be found in the following four factors.

1) Perspective
2) Sample size of good applicants and bad applicants
3) Proportion of good applicants and bad applicants in the population
4) Similarity of attributes between the good applicants and bad applicants

Regarding the first factor, perspective determines which goodness of fit test(s) would be used, for example, whether to use a cut-off at 10%, 20%, K-S statistic, or others. The choice of test depends on the application of the credit scoring model. A fraud model may have a cut-off point of 5%, whereas a credit card application model may have a 30% cut-off. This study aimed to understand modeling technique performance across different perspectives by using various tools for evaluating the models, including cross validation at 5%, 10%, 20%, and 30% cut-off, the K-S statistic, and the Gini coefficient.

The second factor was due to the fact that sample size plays a significant role in all statistical analysis. Hair, Black, Babin, and Anderson (2010) have stated that "…the discussion of statistical power demonstrated the substantial impact sample size plays in achieving statistical significance, both in small and large sample sizes…" and that "…sample sizes affect the results when the analyses involve groups of observations, such as discriminant analysis. Unequal sample sizes among groups influence the results and require additional analysis..." Thus it is important to determine the effect of sample size on the performance of the various modeling techniques. This study strived to understand the modeling technique performance across three sample sizes: balanced samples which had 1,000 good applicants and 1,000 bad applicants; unbalanced samples which had 4,000 good applicants and 1,000 bad applicants; and the more unbalanced samples which had 9,000 good applicants and 1,000 bad applicants. The simulation results in this study have illustrated clearly that sample size does influence credit scoring model performance.

The third factor was to represent the various populations in credit scoring such as varying from prime to sub-prime applicants. When the economy is good, there will be a high proportion of good applicants, whereas when the economy is in recession, the proportion of bad applicants will increase once good applicants have turned bad with the economic times. This study incorporated the effect of the proportion of good applicants and bad applicants in the population on the choice of credit scoring models by creating populations with three different proportions of good applicants and bad applicants. The simulation results in this study also showed that this factor does affect credit scoring model performance.

Finally, the fourth and most important factor was the similarity of the attributes of the good and bad applicants. This factor relates to generalizing across the different values of the independent variables. The variables in credit scoring models will differ, sometimes even significantly. Credit scoring models are used for a variety of applications related to extensions of credit, such as determining whether or not to give a credit card, home loan, auto loan, fraud detection, etc. For example, the variables for a fraud model would include the identification of variables, whereas these variables would not be in a home loan model. It is not just the variables used but also the number of variables in the model that will vary for each application and from financial institution to financial institution. Some of this was seen in the literature review where previous studies on real data varied in variables and in the number of variables. Although the independent variables may differ, some variables are correlated and may even overlap in nature, for example, the number of inquires for credit in the past 12 months and the number of inquires for credit in the past 6 months.

It is impossible to investigate all possible variables and with all possible parameters. Even if it were possible, the person reading the results would have to then align his or her situation with the specific parameters used. Given the latter, this fourth factor is considered a good proxy for the various possibilities of independent variables. In order to obtain the simulated data that is generic and realistic, the distributions and their parameters for the independent variables in the simulation study were taken from Dryver and Jantra Sukkasem (2009).

There are ten independent variables with various distributions. Also the independent variables have varying similarities of parameters from good applicants and bad applicants. In addition, it was assumed that good applicants may have some bad attributes and that bad applicants will have some good attributes, just as in real life, thus adding to the complexity, but more important, to the reality of the simulation. The similarity or lack of similarity between the goods and bads was used as a proxy for obtaining a general view of the effect of the independent variables on the model performance under various scenarios. The objective of any credit scoring model is to differentiate between goods and bads. The fourth factor was incorporated in this study in order to answer how models perform when good applicants and bad applicants look very different versus how they perform when good applicants and bad applicants look very similar. The logic was that varying the distributions in the independent variables affects this part of the modeling by changing the degree of ease (or lack of ease) in distinguishing between good applicants and bad applicants; thus, the assumption that varying the similarity between good applicants and bad applicants serves as a proxy for understanding the scenarios with various attributes of independent variables.

In order to avoid the problem that the models will be specific only to the sample data or "overfit" the sample, this study validated the models by using the population data to test the models. This ensured that the results obtained from the study were generalizable to the population data.

Both linear discriminant analysis and logistic regression utilize maximum likelihood estimation to obtain their parameter estimates. That is why it can be seen in the simulation results that both methods yield very similar results in most scenarios. Nevertheless, the two methods differ in their basic idea. Discriminant analysis relies on the assumptions of multivariate normality and equal variance-covariance matrices across groups, while logistic regression does not face these assumptions and is much more robust when these assumptions are not met. However, if the assumptions of the multivariate normality of the independent variables within each group of the dependant variable are met, and each category has the same variance and covariance for the predictors, the discriminant analysis might provide a more accurate classification (Grimm and Yarnold, 1995; Tabachnick and Fidell, 1996).

Discriminant analysis was achieved by calculating the discriminant function in order to maximize the differences between the groups. Discriminant function produces discriminant Z scores and predicts group membership based on those scores. The discriminant Z scores are the linear combinations of each independent variable and its discriminant weight. They are not bounded by any range. Logistic regression produces the likelihood of each observation being in the group that is coded as "1." The predicted scores from the logistic regression function are bounded by zero and one (Hair et al., 2010; Worth and Cronin, 2003).

From the simulation results, it can be seen that in the scenarios where the relative size of two groups in the sample does affect the relative model performance, logistic regression is superior for the balanced-samples, while discriminant analysis is superior for the unbalanced-samples. Moreover, logistic regression is superior when the good applicants and bad applicants are highly different in attributes, whereas discriminant analysis is superior when the good and bad applicants have similar characteristics.

While logistic regression and discriminant analysis are similar in the sense that they both employ the method of maximum likelihood estimations, the recursive partitioning (decision tree) is different. Recursive partitioning employs the concept of splitting (partitioning) the training set. During the process of decision tree induction, every possible value of every possible feature within the training set represents a potential split that could be used to divide all observations into groups. Any particular node will have at most two paths leading from it to the next node(s) in the path. The result is splitting the data at each node into two independent groups; this is partitioning. Once the two new nodes linked to a previous node are formed, the process is repeated for each new node independently using only the observations present in that node; this is the recursive step. Recursive partitioning splits the observations such that observations with similar response values are grouped. The trees constructed from different samples usually have different numbers of final nodes and each final node also has different sizes. Because all of the observations in the same final node are given the same predicted response value, when using the recursive partitioning approach to predict and rank the credit score of loan applicants, the predicted score at the pre-specified cut-off percentile leads to the rejection of all the other observations that have that same score. This was illustrated in the simulation study—when using the recursive partitioning method, the percentage of rejection of many recursive partitioning models was larger than the pre-specified percentage.

The findings of this simulation study, which are presented in Table 18, can be summarized as follows.

For the high K-S population, three measures, namely Accuracy, Type I error, and Odds ratio at 10% cut-off, were perfectly consistent with those at a 20% cut-off. These three measures were also consistent with the K-S statistic and Gini coefficient. All of the evaluation tools agreed that logistic regression is the best method in predicting good and bad loan applicants.

For the mid K-S population, the three measures at 10% cut-off were consistent with those at a 20% cut-off and with the K-S and Gini for all except in one case: when sample type 2 and 3 were drawn from population type 1, the three measure at a 10% cut-off were inconsistent with the other measures and with both evaluation tools. When sample type 1 was drawn from any population type, all of the measures and evaluation tools agreed that logistic regression was the best method, whereas when sample type 2 and 3 were drawn from any population type, logistic regression as the best method when evaluated at a 10% cut-off; however, discriminant analysis was the best method when evaluated at a 20% cut-off.

For the low K-S population, discriminant analysis was the best method when evaluated at both a 10% and 20% cut-off based on any of the three measures. However, there was a conflict between the three measures and the K-S and Gini when sample type 2 and 3 were drawn from any population. Although the three measures perfectly suggested that discriminant analysis was the best method in all cases, the K-S and Gini suggested that logistic regression was the best method when sample type 2 and 3 are drawn from any population.

With additional assumptions, the economic significance can be investigated directly when comparing the difference in bad rate between the best and the second best model. For example, when observing the bad rate at a 20% cut-off, under the mid

K-S population with a 70:30 good:bad ratio, when the logistic model was formed based on sample type 1 (1,000 goods and 1,000 bads), it yielded an average of 18.64% bad rate; however, the discriminant analysis that was formed based on the same sample yielded an average of 19.05% bad rate. The difference in the bad rate between the two models was 0.41%. If we assume that the bank has a portfolio of ten million personal loans, then it turns out that a loan decision based on the discriminant model will result in accepting 41,000 more bad loans, relative to the logistic model. In monetary terms, if we assume that a loss on one loan is only USD 10,000, then the incremental losses will be USD 410 million. However, if the sample type changes to be sample type 3 (9,000 goods and 1,000 bads), then with the same assumption, the loan decision based on the logistic model, will result in an incremental loss of USD 417 million. These amounts can be considered as having high economic significance.

From these observations, it can be seen clearly that there is no perfect solution to credit scoring. When banks and financial institutions build credit scoring models, they should understand the nature of the population that they have to deal with and the sample sets that they have on hand. Then, based on a particular population and sample set, they can select the statistical method that performs relatively better compared to other methods.

Some practitioners of credit scoring may compare the performance of different credit scoring techniques on their sample sets to determine the optimal technique and use it to construct a credit scoring model. In order to achieve this objective, different techniques are tested based on limited data sets, and ultimately the selected model will be implemented on the population. In this research, a simulation was performed in order to overcome the limitations of data sets so as to achieve a thorough understanding of how different techniques will perform on the population, and how sensitive the relative performance of each technique is to the change in the characteristics of populations and samples. As a result, the use of simulation in this study yielded more insights, and practitioners should consider the methodology applied in this research rather than simply testing multiple models and comparing them on the limited samples.

Some research ideas for the future include testing these models on real data sets, extending the simulation study by including more statistical methods or combining more than one method into one model, or analyzing further why the K-S and Gini are inconsistent with the confusion matrix measures in some scenarios.

In additions, the limitations of the recursive partitioning model found in this study offer opportunity for further investigation. Future studies may consider including different independent variables in order to investigate whether it is possible for the recursive partitioning model to obtain an exact cut-off percentage. More studies can be done on whether there is a significant trade-off between getting the exact percentage cut-off and having an improvement in model performance.

BIBLIOGRAPHY

Abdou, H., Pointon, J., and El-Masry, A. 2008. Neural Nets Versus Conventional Techniques in Credit Scoring in Egyptian Banking. **Expert Systems with Applications.** 35: 1275 – 1292.

Altman, Edward I. 1980. Commercial Bank Lending: Process, Credit Scoring, and Costs of Errors in Lending. **Journal of Financial and Quantitative Analysis.** 15: 813 – 832.

Atiya, Amir F. 2001. Bankruptcy Prediction for Credit Risk Using Neural Networks: a Survey and New Results. **IEEE Transactions on Neural Networks.** 12 (4): 929 – 935.

Baesens, B., Setiono, R., Mues, C., and Vanthienen, J. 2003. Using Neural Network Rule Extraction and Decision Tables for Credit-Risk Evaluation. **Management Science**. 49 (3): 312 – 329.

Bahl, L. R., Brown, P. F., de Sousa, P. V., Mercer, R. L. 1989. A Tree-Based Language Model for Natural Language Speech Recognition. **IEEE Trans. On AS and SP**. 37: 1001 – 1008.

Breiman, Leo et al. 1984. **Classification and Regression Trees**. Boca Raton: Chapman & Hall/CRC.

Chen, X., Rusinko, A. and Young, S. S. 1998. Recursive Partitioning Analysis of a Large Structure-Activity Data Set Using Three-Dimensional Descriptors. **Journal of Chemical Information and Computer Sciences**. 38: 1054 – 1062.

Crawley, Michael J. 2007. **The R Book**. Chichester: Wiley.

Desai, V. S., Crook, J. N., and Overstreet, G. A. 1996. A Comparison of Neural Networks and Linear Scoring Models in the Credit Union Environment. **European Journal of Operational Research**. 95: 24 – 37.

Dryver, A. L. 2011. Focusing on the Lower Scoring Data in Order to Improve Credit Scoring Model Selection. **Advances and Applications in Statistics**. 20 (1): 25 – 41.

Dryver, A. L. and Jantra Sukkasem. 2009. Validating Risk Models with a Focus on Credit Scoring Models. **Journal of Statistical Computation and Simulation**. 79 (2): 181 – 193.

Durand, D. 1941. **Risk Elements in Consumer Installment Financing**. New York, NY: National Bureau of Economic Research.

Eisenbeis, R. A. 1987. A Comparative Analysis of Classification Procedures: Discussion. **Journal of Finance**. 42 (3): 681 – 683.

Elder, J. F. and Pregibon, D. 1996. A Statistical Perspective on Knowledge Discovery in Databases. In **Advances in Knowledge Discovery and Data Mining**. Usama M. Fayyad, ed. Menlo Park, Calif.: AAAI Press/The MIT Press.

Emel, A. B., Oral, M., Reisman, A. and Yolalan, R. 2003. A Credit Scoring Approach for the Commercial Banking Sector. **Socio-Economic Planning Sciences**. 37: 103 – 123.

Fisher, R. A. 1936. The Use of Multiple Measurements in Taxonomic Problems. **Annals of Eqgenics**. 7 (2): 179 – 188.

Flury, Bernhard and Riedwyl, Hans. 1985. T^2 tests, the Linear Two-Group Discriminant Function, and Their Computation by Linear Regression. **American Statistician.** 39 (1): 20 – 25.

Friedman, J. H. 1989. Regularized Discriminant Analysis. **Journal of the American Statistical Association**. 84 (405): 165 – 175.

Galindo, J. and Tamayo, P. 2000. Credit Risk Assessment Using Statistical and Machine Learning: Basic Methodology and Risk Modeling Applications. **Computational Economics**. 15: 107 – 143.

Goldman, L., Cook, F., Johnson, P., Brand , D., Rouan, G. and Lee, T. 1996. Prediction of the Need for Intensive Care in Patients Who Come to Emergency Departments with Acute Chest Pain. **The New England Journal of Medicine**. 334: 1498 – 504.

Goldman, L. et al. 1988. A Computer Protocol to Predict Myocardial Infarction in Emergency Department Patients with Chest Pain. **The New England Journal of Medicine**. 318 (13): 797-803.

Grimm, L. G. and Yarnold, P. R., eds. 1995. **Reading and Understanding Multivariate Statistics**. Washington D.C.: American Psychological Association.

Hair, Joseph F. Jr., Black, William C., Babin, Barry J. and Anderson, Rolph E. 2010. **Multivariate Data Analysis: A Global Perspective**. New Jersey: Pearson.

Hand, D. J. and Henley, W. E. 1996. A K-Nearest-Neighbour Classifier for Assessing Consumer Credit Risk. **Journal of the Royal Statistical Society. Series D (The Statistician)**. 45 (1): 77 – 95.

Hand, D. J. and Henley, W. E. 1997. Statistical Classification Methods in Consumer Credit Scoring: A Review. **Journal of the Royal Statistical Society. Series A (Statistics in Society)**. 160 (3): 523 – 541.

Hastie, Trevor and Tibshirani, Robert. 1996. Discriminant Analysis by Gaussian Mixtures. **Journal of the Royal Statistical Society. Series B (Methodological)**. 58 (1): 155 – 176.

Hosmer, David W. and Lemeshow, Stanley. 2000. **Applied Logistic Regression.** 2nd ed. New York: Wiley.

Hsieh, N. 2005. Hybrid Mining Approach in the Design of Credit Scoring Models. **Expert Systems with Applications**. 28: 655 – 665.

Kolesar, P. and Showers, J. L. 1985. A Robust Credit Screening Model Using Categorical Data. **Management Science**. 31 (2): 123 – 133.

Lawrence, K., Pai, D., Klimberg, R., Kudbya, S. and Lawrence, S. 2010. Segmenting Financial Services Market: An Empirical Study of Statistical and Non-Parametric Methods. In **Handbook of Quantitative Finance and Risk Management.** Cheng-Few Lee, Alice C. Lee and John Lee, eds. New York: Springer. Pp. 1061-1066.

Mayers, J. H. and Forgy, E. W. 1963. The Development of Numerical Credit Evaluation Systems. **Journal of American Statistics Association**. 58: 799 – 806.

Mester, L. J. 1997. What's the Point of Credit Scoring?. **Federal Reserve Bank of Philadelphia Business Review**. 3: 3 – 16.

Orgler, Y. E. 1970. A Credit Scoring Model for Commercial Loans. **Journal of Money, Credit and Banking**. 2 (4): 435 – 445.

Owens, E. A., Griffiths, R. E. and Ratnatunga, K. U. 1996. Using Oblique Decision Trees for the Morphological Classification of Galaxies. **Monthly Notices of the Royal Astronomical Society**. 281: 153 – 157.

Press, S. J. and Wilson, S. 1978. Choosing Between Logistic Regression and Discriminant Analysis. **Journal of the American Statistical Association**. 73 (364): 699 – 705.

Press, William H., Teukolsky, Saul A., Vetterling, William T. and Flannery, Brian P. 1992. **Numerical Recipes in C: The Art of Scientific Computing**. Cambridge: Cambridge University Press.

Quinlan, J. R. 1987. Generating Production Rules from Decision Trees. In **Proceedings of the Tenth International Joint Conference on Artificial Intelligence**. San Francisco, CA: Morgan Kaufmann. Pp. 304-307.

Reichert, A. K., Cho, C. and Wagner, G. M. 1983. An Examination of the Conceptual Issues Involved in Developing Credit-Scoring Models. **Journal of Business & Economic Statistics**. 1 (2): 101 – 114.

Ripley, B. D. 1994. Neural Networks and Related Methods for Classification (with Discussion). **Journal of Research Statistics Society. Series B (Methodological)**. 56: 409 – 456.

Rosenberg, E. and Gleit, A. 1994. Quantitative Methods in Credit Management: A Survey. **Operations Research**. 42: 589 – 613.

Siddiqi, N. 2006. **Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring**. Hoboken, New Jersey: John Wiley and Sons, Inc.

Srinivasan, V. and Kim, Y. H. 1987. Credit Granting: A Comparative Analysis of Classification Procedures. **Journal of Finance**. 42 (3): 665 – 681.

Tabachnick, B. G. and Fidell, L. S. 1996. **Using Multivariate Statistics**. NY: HarperCollins.

Thomas, L. C., Edelman, D. B. and Crook, J. N. 2002. **Credit Scoring and Its Applications**. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Trippi, R. R. and Turban, E. 1993. **Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real-World Performance**. Singapore: Heinemann Asia.

West, D. 2000. Neural Network Credit Scoring Models. **Computers & Operations Research**. 27 (11 – 12): 1131 – 1152.

Wiginton, J. C. 1980. A Note on the Comparision of Logit and Discriminant Models of Consumer Credit Behavior. **Journal of Financial Quantitative Analysis**. 15: 757 – 770.

Worth, A. P. and Cronin, M. T. D. 2003. The Use of Discriminant Analysis, Logistic Regression and Classification Tree Analysis in the Development of Classification Models for Human Health Effects. **Theochem**. 622: 97-111.

Zhang, H. P. 2004. Recursive Partitioning and Tree-Based Methods. In **Handbook of Computational Statistics**. J. E. Gentle, W. Härdle and Y. Mori, eds. Berlin: Springer. Pp. 813-840.

Zhang, H. P., Crowley, J., Sox, H. and Olshen, R. A. 2001. Tree Structural Statistical Methods. In **Encyclopedia of Biostatistics**. Vol. 6. Chichester, West Sussex: Wiley. Pp. 4561-4573.

Zhang, H. P. and Singer, B. 1999. **Recursive Partitioning in the Health Sciences**. New York: Springer Verlag.

APPENDICES

Appendix A

The R Codes

1.1 Setting the Parameters for Each Population Type

1.1.1 pop7010.R

```
tg=700000                          # obs of good (row)
tb=300000                          # obs of bad (row)
n=1                                # iterations (column)
```

1.1.2 pop8020.R

```
tg=800000                          # obs of good (row)
tb=200000                          # obs of bad (row)
n=1                                # iterations (column)
```

1.1.3 pop9010.R

```
tg=900000                          # obs of good (row)
tb=100000                          # obs of bad (row)
n=1                                # iterations (column)
```

1.1.4 ks_high.R

```
PIG=0.1
PIB=0.83
LAM1G=2
LAM1B=3.5
LAM2G=3.5
LAM2B=8
LAM3G=12
LAM3B=16
PI4G=0.2
PI4B=0.6
```

```
PI5G=0.04
PI5B=0.2
PI6G=0.2
PI6B=0.85
LAM7G=12
LAM7B=16
PI8G=0.3
PI8B=0.6
MU9G=5080
MU9B=5040
PI10G=0.4
PI10B=0.7
```

1.1.5 ks_mid.R

```
PIG=0.1
PIB=0.775
LAM1G=2
LAM1B=2.5
LAM2G=4.5
LAM2B=6
LAM3G=13.5
LAM3B=15
PI4G=0.2
PI4B=0.5
PI5G=0.04
PI5B=0.15
PI6G=0.25
PI6B=0.55
LAM7G=5
LAM7B=30
PI8G=0.3
PI8B=0.5
```

```
MU9G=5050
MU9B=5020
PI10G=0.23
PI10B=0.45
```

1.1.6 ks_low.R

```
PIG=0.1
PIB=0.7
LAM1G=2
LAM1B=2.5
LAM2G=5
LAM2B=6
LAM3G=13
LAM3B=14
PI4G=0.2
PI4B=0.5
PI5G=0.04
PI5B=0.09
PI6G=0.05
PI6B=0.15
LAM7G=25
LAM7B=30
PI8G=0.3
PI8B=0.5
MU9G=5030
MU9B=5018
PI10G=0.4
PI10B=0.5
```

1.2 Generating the Attributes of Populations (1_pop_gen.R)

```
yg =matrix(1,tg,n)                          # all one
zg1=matrix( rbinom(tg*n,1,PIG),tg,n )
zg2=matrix( rbinom(tg*n,1,PIG),tg,n )
zg3=matrix( rbinom(tg*n,1,PIG),tg,n )
zg4=matrix( rbinom(tg*n,1,PIG),tg,n )
zg5=matrix( rbinom(tg*n,1,PIG),tg,n )
zg6=matrix( rbinom(tg*n,1,PIG),tg,n )
zg7=matrix( rbinom(tg*n,1,PIG),tg,n )
zg8=matrix( rbinom(tg*n,1,PIG),tg,n )
zg9=matrix( rbinom(tg*n,1,PIG),tg,n )
zg10=matrix( rbinom(tg*n,1,PIG),tg,n )

# The good attributes

 xg1=matrix( zg1*rpois(tg*n,LAM1G)+(1-zg1)*rpois(tg*n,LAM1B)+1,tg,n )
 xg2=matrix( zg2*rpois(tg*n,LAM2G)+(1-zg2)*rpois(tg*n,LAM2B)+1,tg,n )
 xg3=matrix( zg3*rpois(tg*n,LAM3G)+(1-zg3)*rpois(tg*n,LAM3B)+xg2,tg,n )
 xg4=matrix( zg4*(runif(tg*n,min=0,max=1)+PI4G)/(1+PI4G)+(1-zg4)*(runif
(tg*n,min=0,max=1)+PI4B)/(1+PI4B),tg,n)
 xg5=matrix( zg5*rbinom(tg*n,1,PI5G)+(1-zg5)*rbinom(tg*n,1,PI5B),tg,n )
xg6=matrix( zg6*rbinom(tg*n,1,PI6G)+(1-zg6)*rbinom(tg*n,1,PI6B),tg,n )
 xg7=matrix( zg7* rexp(tg*n,rate=LAM7G) + (1-zg7)* rexp(tg*n,rate=LAM7B),tg,n)
 xg8=matrix( zg8*rbinom(tg*n,1,PI8G) + (1-zg8)*rbinom(tg*n,1,PI8B),tg,n )
 xg9=matrix( zg9*rnorm(tg*n,mean=MU9G,sd=30)+(1-zg9)*rnorm
(tg*n,mean=MU9B,sd=30),tg,n)
xg10=matrix( zg10*rbinom(tg*n,1,PI10G) + (1-zg10)*rbinom(tg*n,1,PI10B),tg,n )

yb =matrix(0,tb,n)                          # all zero
zb1=matrix( rbinom(tb*n,1,PIB),tb,n )
zb2=matrix( rbinom(tb*n,1,PIB),tb,n )
```

```
zb3=matrix( rbinom(tb*n,1,PIB),tb,n )
zb4=matrix( rbinom(tb*n,1,PIB),tb,n )
zb5=matrix( rbinom(tb*n,1,PIB),tb,n )
zb6=matrix( rbinom(tb*n,1,PIB),tb,n )
zb7=matrix( rbinom(tb*n,1,PIB),tb,n )
zb8=matrix( rbinom(tb*n,1,PIB),tb,n )
zb9=matrix( rbinom(tb*n,1,PIB),tb,n )
zb10=matrix( rbinom(tb*n,1,PIB),tb,n )

# The bad attributes

xb1=matrix( zb1*rpois(tb*n,LAM1G)+(1-zb1)*rpois(tb*n,LAM1B)+1,tb,n )
xb2=matrix( zb2*rpois(tb*n,LAM2G)+(1-zb2)*rpois(tb*n,LAM2B)+1,tb,n )
xb3=matrix( zb3*rpois(tb*n,LAM3G)+(1-zb3)*rpois(tb*n,LAM3B)+ xb2 ,tb,n )
xb4=matrix( zb4*(runif(tb*n,min=0,max=1)+PI4G)/(1+PI4G)+(1-
zb4)*(runif(tb*n,min=0,max=1)+PI4B)/(1+PI4B),tb,n)
xb5=matrix( zb5*rbinom(tb*n,1,PI5G)+(1-zb5)*rbinom(tb*n,1,PI5B),tb,n )
xb6=matrix( zb6*rbinom(tb*n,1,PI6G)+(1-zb6)*rbinom(tb*n,1,PI6B),tb,n )
xb7=matrix( zb7* rexp(tb*n,rate=LAM7G) + (1-zb7)* rexp(tb*n,rate=LAM7B),tb,n )
xb8=matrix( zb8*rbinom(tb*n,1,PI8G) + (1-zb8)*rbinom(tb*n,1,PI8B),tb,n )
xb9=matrix( zb9*rnorm(tb*n,mean=MU9G,sd=30)+(1-
zb9)*rnorm(tb*n,mean=MU9B,sd=30),tb,n)
xb10=matrix( zb10*rbinom(tb*n,1,PI10G) + (1-zb10)*rbinom(tb*n,1,PI10B),tb,n )

pop.g=cbind(yg,xg1,xg2,xg3,xg4,xg5,xg6,xg7,xg8,xg9,xg10)
#index.g=matrix(c(1:tg),tg,n)
#pop.good=cbind(index.g,pop.g)

pop.b=cbind(yb,xb1,xb2,xb3,xb4,xb5,xb6,xb7,xb8,xb9,xb10)
#index.b=matrix(c(1:tb),tb,n)
#pop.bad=cbind(index.b,pop.b)
```

1.3 Simulating Nine Population Sets (1010.R)

1.3.1 Population with 700,000 goods and 300,000 bads, high K-S

```
memory.limit(4095)
rm(list=ls(all=TRUE))
setwd("D:\\R\\")
library(abind)

source(file="pop7030.R")
source(file="ks_high.R")

source(file="1_pop_gen.R")
Apop=abind(pop.b,pop.g,along=1)

write.table(Apop, file = "70H_Apop.csv",quote=TRUE, sep = ",",
col.names=TRUE, row.names=TRUE, qmethod="double")
```

1.3.2 Population with 700,000 goods and 300,000 bads, mid K-S

```
memory.limit(4095)
rm(list=ls(all=TRUE))
setwd("D:\\R\\")
library(abind)

source(file="pop7030.R")
source(file="ks_mid.R")

source(file="1_pop_gen.R")
Apop=abind(pop.b,pop.g,along=1)

write.table(Apop, file = "70M_Apop.csv",quote=TRUE, sep = ",",
col.names=TRUE, row.names=TRUE, qmethod="double")
```

1.3.3 Population with 700,000 goods and 300,000 bads, low K-S

```
memory.limit(4095)
rm(list=ls(all=TRUE))
setwd("D:\\R\\")
library(abind)

source(file="pop7030.R")
source(file="ks_low.R")

source(file="1_pop_gen.R")
Apop=abind(pop.b,pop.g,along=1)

write.table(Apop, file = "70L_Apop.csv",quote=TRUE, sep = ",",
col.names=TRUE, row.names=TRUE, qmethod="double")
```

1.3.4 Population with 800,000 goods and 200,000 bads, high K-S

```
memory.limit(4095)
rm(list=ls(all=TRUE))
setwd("D:\\R\\")
library(abind)

source(file="pop8020.R")
source(file="ks_high.R")

source(file="1_pop_gen.R")
Apop=abind(pop.b,pop.g,along=1)

write.table(Apop, file = "80H_Apop.csv",quote=TRUE, sep = ",",
col.names=TRUE, row.names=TRUE, qmethod="double")
```

1.3.5 Population with 800,000 goods and 200,000 bads, mid K-S

```
memory.limit(4095)
```

```
rm(list=ls(all=TRUE))
setwd("D:\\R\\")
library(abind)

source(file=" pop8020.R")
source(file="ks_mid.R")

source(file="1_pop_gen.R")
Apop=abind(pop.b,pop.g,along=1)

write.table(Apop, file = "80M_Apop.csv",quote=TRUE, sep = ",",
col.names=TRUE, row.names=TRUE, qmethod="double")
```

1.3.6 Population with 800,000 goods and 200,000 bads, low K-S

```
memory.limit(4095)
rm(list=ls(all=TRUE))
setwd("D:\\R\\")
library(abind)

source(file=" pop8020.R")
source(file="ks_low.R")

source(file="1_pop_gen.R")
Apop=abind(pop.b,pop.g,along=1)

write.table(Apop, file = "80L_Apop.csv",quote=TRUE, sep = ",",
col.names=TRUE, row.names=TRUE, qmethod="double")
```

1.3.7 Population with 900,000 goods and 100,000 bads, high K-S

```
memory.limit(4095)
rm(list=ls(all=TRUE))
setwd("D:\\R\\")
```

```
library(abind)

source(file="pop9010.R")
source(file="ks_high.R")

source(file="1_pop_gen.R")
Apop=abind(pop.b,pop.g,along=1)

write.table(Apop, file = "90H_Apop.csv",quote=TRUE, sep = ",",
col.names=TRUE, row.names=TRUE, qmethod="double")
```

1.3.8 Population with 900,000 goods and 100,000 bads, mid K-S

```
memory.limit(4095)
rm(list=ls(all=TRUE))
setwd("D:\\R\\")
library(abind)

source(file=" pop9010.R")
source(file="ks_mid.R")

source(file="1_pop_gen.R")
Apop=abind(pop.b,pop.g,along=1)

write.table(Apop, file = "90M_Apop.csv",quote=TRUE, sep = ",",
col.names=TRUE, row.names=TRUE, qmethod="double")
```

1.3.9 Population with 900,000 goods and 100,000 bads, low K-S

```
memory.limit(4095)
rm(list=ls(all=TRUE))
setwd("D:\\R\\")
library(abind)
```

```
source(file=" pop9010.R")
source(file="ks_low.R")

source(file="1_pop_gen.R")
Apop=abind(pop.b,pop.g,along=1)

write.table(Apop, file = "90L_Apop.csv",quote=TRUE, sep = ",",
col.names=TRUE, row.names=TRUE, qmethod="double")
```

1.4 Sampling and Iterations

1.4.1 Sample with 1,000 goods and 1,000 bads (sample1.R)

```
mg=1000
mb=1000
```

1.4.2 Sample with 4,000 goods and 1,000 bads (sample2.R)

```
mg=4000
mb=1000
```

1.4.3 Sample with 9,000 goods and 1,000 bads (sample3.R)

```
mg=9000
mb=1000
```

1.4.4 Sampling, estimating the models, and testing the models (loop.R)

```
whb=sample(idb,mb)
whg=sample(idg,mg)

A=abind(Apop[whb,],Apop[whg,],along=1)
ID=c(whb,whg)
A=data.frame(ID,A)
```

```
colnames(A)=c("ID","Y","x1","x2","x3","x4","x5","x6","x7","x8","x9","x10")
rownames(A)=c(1:(mb+mg))

# Use the sample to form LR, LG, RP1, and RP2

LR=glm(Y ~ x1+x2+x3+x4+x5+x6+x7+x8+x9+x10, data=A ) # sample
LG=glm(Y ~ x1+x2+x3+x4+x5+x6+x7+x8+x9+x10, data=A ,
family=binomial(link="logit") ) # sample

RP1=rpart(Y ~ x1+x2+x3+x4+x5+x6+x7+x8+x9+x10, data=A ) # sample
RP2=rpart(Y ~ x1+x2+x3+x4+x5+x6+x7+x8+x9+x10, data=A,
control=rpart.control(cp=0) ) # sample

# Predict population

RPpredict1=predict(RP1,new=list(x1=Apop[,2],x2=Apop[,3],x3=Apop[,4],x4=Apop[
,5],x5=Apop[,6],x6=Apop[,7],x7=Apop[,8],x8=Apop[,9],x9=Apop[,10],x10=Apop[,1
1]))

RPpredict1=as.data.frame(RPpredict1)

ksRP1 =ks.test(RPpredict1[1:tb,],RPpredict1[(tb+1):(tb+tg),])[[1]][[1]]
aucRP1=roc.area(Apop[,1],RPpredict1[,1])$A

B1=data.frame(RPpredict1,Apop[,1])
colnames(B1)=c("YhatRP1","Y")
B1=B1[order(B1$YhatRP1),]

RPpredict2=predict(RP2,new=list(x1=Apop[,2],x2=Apop[,3],x3=Apop[,4],x4=Apop[
,5],x5=Apop[,6],x6=Apop[,7],x7=Apop[,8],x8=Apop[,9],x9=Apop[,10],x10=Apop[,1
1]))
```

```
RPpredict2=as.data.frame(RPpredict2)

ksRP2 =ks.test(RPpredict2[1:tb,],RPpredict2[(tb+1):(tb+tg),])[[1]][[1]]
aucRP2=roc.area(Apop[,1],RPpredict2[,1])$A

B2=data.frame(RPpredict2,Apop[,1])
colnames(B2)=c("YhatRP2","Y")
B2=B2[order(B2$YhatRP2),]

# RP1 at 5% cut off point

Yhat1=B1
score=Yhat1[0.05*(tg+tb),1]
Yhat1$YhatRP1[Yhat1$YhatRP1>score]=1
Yhat1$YhatRP1[Yhat1$YhatRP1<=score]=0
R105=tg+tb-sum(Yhat1$YhatRP1)
GR105=sum(Yhat1$Y[1:R105])
rm(Yhat1,score)

# RP1 at 10% cut off point

Yhat1=B1
score=Yhat1[0.1*(tg+tb),1]
Yhat1$YhatRP1[Yhat1$YhatRP1>score]=1
Yhat1$YhatRP1[Yhat1$YhatRP1<=score]=0
R110=tg+tb-sum(Yhat1$YhatRP1)
GR110=sum(Yhat1$Y[1:R110])
rm(Yhat1,score)

# RP1 at 20% cut off point

Yhat1=B1
```

```
score=Yhat1[0.2*(tg+tb),1]
Yhat1$YhatRP1[Yhat1$YhatRP1>score]=1
Yhat1$YhatRP1[Yhat1$YhatRP1<=score]=0
R120=tg+tb-sum(Yhat1$YhatRP1)
GR120=sum(Yhat1$Y[1:R120])
rm(Yhat1,score)

# RP1 at 30% cut off point

Yhat1=B1
score=Yhat1[0.3*(tg+tb),1]
Yhat1$YhatRP1[Yhat1$YhatRP1>score]=1
Yhat1$YhatRP1[Yhat1$YhatRP1<=score]=0
R130=tg+tb-sum(Yhat1$YhatRP1)
GR130=sum(Yhat1$Y[1:R130])
rm(Yhat1,score)

# RP2 at 5% cut off point

Yhat2=B2
score=Yhat2[0.05*(tg+tb),1]
Yhat2$YhatRP2[Yhat2$YhatRP2>score]=1
Yhat2$YhatRP2[Yhat2$YhatRP2<=score]=0
R205=tg+tb-sum(Yhat2$YhatRP2)
GR205=sum(Yhat2$Y[1:R205])
rm(Yhat2,score)

# RP2 at 10% cut off point

Yhat2=B2
score=Yhat2[0.1*(tg+tb),1]
Yhat2$YhatRP2[Yhat2$YhatRP2>score]=1
```

```
Yhat2$YhatRP2[Yhat2$YhatRP2<=score]=0
R210=tg+tb-sum(Yhat2$YhatRP2)
GR210=sum(Yhat2$Y[1:R210])
rm(Yhat2,score)

# RP2 at 20% cut off point

Yhat2=B2
score=Yhat2[0.2*(tg+tb),1]
Yhat2$YhatRP2[Yhat2$YhatRP2>score]=1
Yhat2$YhatRP2[Yhat2$YhatRP2<=score]=0
R220=tg+tb-sum(Yhat2$YhatRP2)
GR220=sum(Yhat2$Y[1:R220])
rm(Yhat2,score)

# RP2 at 30% cut off point

Yhat2=B2
score=Yhat2[0.3*(tg+tb),1]
Yhat2$YhatRP2[Yhat2$YhatRP2>score]=1
Yhat2$YhatRP2[Yhat2$YhatRP2<=score]=0
R230=tg+tb-sum(Yhat2$YhatRP2)
GR230=sum(Yhat2$Y[1:R230])
rm(Yhat2,score)


# outputs for all

coefLR_i=data.frame(LR[[1]][[1]],LR[[1]][[2]],LR[[1]][[3]],LR[[1]][[4]],LR[[1]][[5]
],LR[[1]][[6]],LR[[1]][[7]],LR[[1]][[8]],LR[[1]][[9]],LR[[1]][[10]],LR[[1]][[11]])
coefLG_i=data.frame(LG[[1]][[1]],LG[[1]][[2]],LG[[1]][[3]],LG[[1]][[4]],LG[[1]][[5
]],LG[[1]][[6]],LG[[1]][[7]],LG[[1]][[8]],LG[[1]][[9]],LG[[1]][[10]],LG[[1]][[11]])
```

```
evaRPi=data.frame(ksRP1,aucRP1,ksRP2,aucRP2,GR105,GR110,GR120,GR130,GR
205,GR210,GR220,GR230,R105,R110,R120,R130,R205,R210,R220,R230)

coefLR=rbind(coefLR,coefLR_i)
coefLG=rbind(coefLG,coefLG_i)

evaRP=rbind(evaRP,evaRPi)
```

1.4.5 Removing the Previous Iteration (remove.R)

```
        rm(whb,whg,A,ID,LR,LG,RP1,RP2,RPpredict1,RPpredict2,ksRP1,ksR
P2,aucRP1,aucRP2,B1,B2)

        rm(GR105,GR110,GR120,GR130,GR205,GR210,GR220,GR230,R105,
R110,R120,R130,R205,R210,R220,R230,coefLR_i,coefLG_i,evaRPi)
```

1.4.6 Sampling and Iterations

```
        memory.limit(4095)
        setwd("D:\\RJib\\")     # change file name to where you save the files
        library(abind)
        library(waveslim)
        library(spam)
        library(fields)
        library(boot)
        library(MASS)
        library(CircStats)
        library(verification)
        library("rpart")

Apop=read.csv(file="70H_Apop.csv", head = TRUE, sep =",")
# change file name to 70H, 70M, 70L, 80H, 80M, 80L, 90H, 90M, 90L
```

```
source(file="pop7030.R")                        #change to pop8020, pop9010
source(file="sample1.R")                        # change to sample2, sample3

idb=c(1:tb)
idg=c((tb+1):(tb+tg))

coefLR=data.frame()
coefLG=data.frame()
evaRP=data.frame()

source(file="loop.R")
source(file="remove.R")

# repeat line "loop.R" and "remove.R" for 1,000 iterations
# do this to avoid the loop command below
# for (i in 1:1000) {
#       source(file="loop.R")
#       source(file="remove.R")
#       }

write.table(coefLR, file = "coefLR.csv",append=FALSE,quote=TRUE, sep = ",",
col.names = TRUE,row.names=TRUE,qmethod="double")

write.table(coefLG, file = "coefLG.csv",append=FALSE,quote=TRUE, sep = ",",
col.names = TRUE,row.names=TRUE,qmethod="double")

write.table(evaRP, file = "evaRP.csv",append=FALSE,quote=TRUE, sep = ",",
col.names = TRUE,row.names=TRUE,qmethod="double")

# repeat this subsection for each population type and each sample type
```

1.5 The Codes for Linear Regression, Recursive Partitioning, K-S, and Area Under ROC

1.5.1 Linear Regression (GLM)

```
function (formula, family = gaussian, data, weights, subset,
na.action, start = NULL, etastart, mustart, offset, control = list(...),
model = TRUE, method = "glm.fit", x = FALSE, y = TRUE, contrasts =
NULL,
   ...)
{
call <- match.call()
if (is.character(family))
family <- get(family, mode = "function", envir = parent.frame())
if (is.function(family))
family <- family()
if (is.null(family$family)) {
print(family)
stop("'family' not recognized")
}
if (missing(data))
data <- environment(formula)
mf <- match.call(expand.dots = FALSE)
m <- match(c("formula", "data", "subset", "weights", "na.action",
"etastart", "mustart", "offset"), names(mf), 0L)
mf <- mf[c(1L, m)]
mf$drop.unused.levels <- TRUE
mf[[1L]] <- as.name("model.frame")
mf <- eval(mf, parent.frame())
if (identical(method, "model.frame"))
   return(mf)
if (!is.character(method) && !is.function(method))
   stop("invalid 'method' argument")
```

```
if (identical(method, "glm.fit"))
  control <- do.call("glm.control", control)
mt <- attr(mf, "terms")
Y <- model.response(mf, "any")
if (length(dim(Y)) == 1L) {
nm <- rownames(Y)
dim(Y) <- NULL
if (!is.null(nm))
names(Y) <- nm
}
X <- if (!is.empty.model(mt))
model.matrix(mt, mf, contrasts)
else matrix(, NROW(Y), 0L)
weights <- as.vector(model.weights(mf))
if (!is.null(weights) && !is.numeric(weights))
stop("'weights' must be a numeric vector")
if (!is.null(weights) && any(weights < 0))
stop("negative weights not allowed")
offset <- as.vector(model.offset(mf))
if (!is.null(offset)) {
if (length(offset) != NROW(Y))
stop(gettextf("number of offsets is %d should equal %d (number of
observations)",
length(offset), NROW(Y)), domain = NA)
}
mustart <- model.extract(mf, "mustart")
etastart <- model.extract(mf, "etastart")
fit <- eval(call(if (is.function(method)) "method" else method,
x = X, y = Y, weights = weights, start = start, etastart = etastart,
mustart = mustart, offset = offset, family = family,
control = control, intercept = attr(mt, "intercept") >
0L))
```

```
if (length(offset) && attr(mt, "intercept") > 0L) {
fit$null.deviance <- eval(call(if (is.function(method)) "method" else
method,
x = X[, "(Intercept)", drop = FALSE], y = Y, weights = weights,
offset = offset, family = family, control = control,
intercept = TRUE))$deviance
}
if (model)
fit$model <- mf
fit$na.action <- attr(mf, "na.action")
if (x)
fit$x <- X
if (!y)
fit$y <- NULL
fit <- c(fit, list(call = call, formula = formula, terms = mt,
data = data, offset = offset, control = control, method = method,
contrasts = attr(X, "contrasts"), xlevels = .getXlevels(mt,
mf)))
class(fit) <- c(fit$class, c("glm", "lm"))
fit
}
<environment: namespace:stats>
```

1.5.2 Recursive Partitioning (RPART)

```
function (formula, data, weights, subset, na.action = na.rpart,
method, model = FALSE, x = FALSE, y = TRUE, parms, control,
cost, ...)
{
call <- match.call()
if (is.data.frame(model)) {
m <- model
model <- FALSE
```

```
}
else {
m <- match.call(expand.dots = FALSE)
m$model <- m$method <- m$control <- NULL
m$x <- m$y <- m$parms <- m$... <- NULL
m$cost <- NULL
m$na.action <- na.action
m[[1L]] <- as.name("model.frame")
m <- eval(m, parent.frame())
}
Terms <- attr(m, "terms")
if (any(attr(Terms, "order") > 1L))
stop("Trees cannot handle interaction terms")
Y <- model.extract(m, "response")
wt <- model.extract(m, "weights")
if (length(wt) == 0L)
wt <- rep(1, nrow(m))
offset <- attr(Terms, "offset")
X <- rpart.matrix(m)
nobs <- nrow(X)
nvar <- ncol(X)
if (missing(method)) {
if (is.factor(Y) || is.character(Y))
method <- "class"
else if (inherits(Y, "Surv"))
method <- "exp"
else if (is.matrix(Y))
method <- "poisson"
else method <- "anova"
}
if (is.list(method)) {
mlist <- method
```

```
method <- "user"
if (missing(parms))
init <- mlist$init(Y, offset, wt = wt)
else init <- mlist$init(Y, offset, parms, wt)
method.int <- 4L
keep <- rpartcallback(mlist, nobs, init)
}
else {
method.int <- pmatch(method, c("anova", "poisson", "class",
"exp"))
if (is.na(method.int))
stop("Invalid method")
method <- c("anova", "poisson", "class", "exp")[method.int]
if (method.int == 4L)
method.int <- 2L
if (missing(parms))
init <- (get(paste("rpart", method, sep = ".")))(Y,
offset, , wt)
else init <- (get(paste("rpart", method, sep = ".")))(Y,
offset, parms, wt)
ns <- asNamespace("rpart")
if (!is.null(init$print))
environment(init$print) <- ns
if (!is.null(init$summary))
environment(init$summary) <- ns
if (!is.null(init$text))
environment(init$text) <- ns
}
Y <- init$y
xlevels <- attr(X, "column.levels")
cats <- rep(0, ncol(X))
if (!is.null(xlevels)) {
        cats[match(names(xlevels), dimnames(X)[[2]])] <-
unlist(lapply(xlevels,
        length))
        }
        extraArgs <- list(...)
        if (length(extraArgs)) {
        controlargs <- names(formals(rpart.control))
        indx <- match(names(extraArgs), controlargs, nomatch = 0)
        if (any(indx == 0))
        stop("Argument ", names(extraArgs)[indx == 0], "not matched")
        }
        controls <- rpart.control(...)
        if (!missing(control))
        controls[names(control)] <- control
        xval <- controls$xval
        if (is.null(xval) || (length(xval) == 1L && xval == 0) ||
        method == "user") {
        xgroups <- 0
        xval <- 0
        }
        else if (length(xval) == 1L) {
        xgroups <- sample(rep(1:xval, length = nobs), nobs, replace = FALSE)
        }
        else if (length(xval) == nobs) {
        xgroups <- xval
        xval <- length(unique(xgroups))
        }
        else {
        if (!is.null(attr(m, "na.action"))) {
        temp <- as.integer(attr(m, "na.action"))
        xval <- xval[-temp]
        if (length(xval) == nobs) {
```

```
xgroups <- xval
xval <- length(unique(xgroups))
}
else stop("Wrong length for xval")
}
else stop("Wrong length for xval")
}
if (missing(cost))
cost <- rep(1, nvar)
else {
if (length(cost) != nvar)
stop("Cost vector is the wrong length")
if (any(cost <= 0))
stop("Cost vector must be positive")
}
tfun <- function(x) {
if (is.matrix(x))
rep(is.ordered(x), ncol(x))
else is.ordered(x)
}
isord <- unlist(lapply(m[attr(Terms, "term.labels")], tfun))
rpfit <- .C(C_s_to_rp, n = as.integer(nobs), nvarx = as.integer(nvar),
ncat = as.integer(cats * (!isord)), method = as.integer(method.int),
as.double(unlist(controls)), parms = as.double(unlist(init$parms)),
as.integer(xval), as.integer(xgroups), as.double(t(init$y)),
as.double(X), as.integer(!is.finite(X)), error = character(1),
wt = as.double(wt), as.integer(init$numy), as.double(cost),
NAOK = TRUE)
if (rpfit$n == -1)
stop(rpfit$error)
nodes <- rpfit$n
nsplit <- rpfit$nvarx
```

```
numcp <- rpfit$method
ncat <- rpfit$ncat[1]
numresp <- init$numresp
if (nsplit == 0)
xval <- 0
cpcol <- if (xval > 0 && nsplit > 0)
5L
else 3L
if (ncat == 0)
catmat <- 0
else catmat <- matrix(integer(1), ncat, max(cats))
rp <- .C(C_s_to_rp2, as.integer(nobs), as.integer(nsplit),
as.integer(nodes), as.integer(ncat), as.integer(cats *
 (!isord)), as.integer(max(cats)), as.integer(xval),
which = integer(nobs), cptable = matrix(double(numcp *
cpcol), nrow = cpcol), dsplit = matrix(double(1),
nsplit, 3), isplit = matrix(integer(1), nsplit, 3),
csplit = catmat, dnode = matrix(double(1), nodes, 3 +
numresp), inode = matrix(integer(1), nodes, 6))
tname <- c("<leaf>", dimnames(X)[[2]])
if (cpcol == 3)
temp <- c("CP", "nsplit", "rel error")
else temp <- c("CP", "nsplit", "rel error", "xerror", "xstd")
dimnames(rp$cptable) <- list(temp, 1L:numcp)
dn1 <- if (nsplit == 0L)
character(0L)
else tname[rp$isplit[, 1L] + 1L]
splits <- matrix(c(rp$isplit[, 2L:3L], rp$dsplit), ncol = 5L,
dimnames = list(dn1, c("count", "ncat", "improve", "index",
"adj")))
index <- rp$inode[, 2]
nadd <- sum(isord[rp$isplit[, 1L]])
```

```
if (nadd > 0) {
newc <- matrix(integer(1), nadd, max(cats))
cvar <- rp$isplit[, 1L]
indx <- isord[cvar]
cdir <- splits[indx, 2L]
ccut <- floor(splits[indx, 4L])
splits[indx, 2L] <- cats[cvar[indx]]
splits[indx, 4L] <- ncat + 1L:nadd
for (i in 1L:nadd) {
newc[i, 1L:(cats[(cvar[indx])[i]])] <- -1 * as.integer(cdir[i])
newc[i, 1L:ccut[i]] <- as.integer(cdir[i])
}
if (ncat == 0)
catmat <- newc
else catmat <- rbind(rp$csplit, newc)
ncat <- ncat + nadd
}
else catmat <- rp$csplit
if (nsplit == 0) {
frame <- data.frame(row.names = 1, var = "<leaf>", n = rp$inode[,
5L], wt = rp$dnode[, 3L], dev = rp$dnode[, 1L], yval = rp$dnode[,
4L], complexity = rp$dnode[, 2L], ncompete = pmax(0L,
rp$inode[, 3L] - 1L), nsurrogate = rp$inode[, 4L])
}
else {
temp <- ifelse(index == 0, 1, index)
svar <- ifelse(index == 0, 0, rp$isplit[temp, 1L])
frame <- data.frame(row.names = rp$inode[, 1L], var = factor(svar,
0:ncol(X), tname), n = rp$inode[, 5L], wt = rp$dnode[,
3L], dev = rp$dnode[, 1L], yval = rp$dnode[, 4L],
complexity = rp$dnode[, 2L], ncompete = pmax(0L,
rp$inode[, 3L] - 1L), nsurrogate = rp$inode[,
4L])
}
if (method.int == 3L) {
numclass <- init$numresp - 1L
temp <- rp$dnode[, -(1L:4L), drop = FALSE] %*%
diag(init$parms$prior *
sum(init$counts)/pmax(1, init$counts))
yprob <- temp/rowSums(temp)
yval2 <- matrix(rp$dnode[, -(1L:3L)], ncol = numclass +
1)
frame$yval2 <- cbind(yval2, yprob)
}
else if (init$numresp > 1L)
frame$yval2 <- rp$dnode[, -(1L:3L), drop = FALSE]
if (is.null(init$summary))
stop("Initialization routine is missing the summary function")
if (is.null(init$print))
functions <- list(summary = init$summary)
else functions <- list(summary = init$summary, print = init$print)
if (!is.null(init$text))
functions <- c(functions, list(text = init$text))
if (method == "user")
functions <- c(functions, mlist)
where <- rp$which
names(where) <- row.names(m)
if (nsplit == 0L) {
ans <- list(frame = frame, where = where, call = call,
terms = Terms, cptable = t(rp$cptable), method = method,
parms = init$parms, control = controls, functions = functions)
}
else {
ans <- list(frame = frame, where = where, call = call,
terms = Terms, cptable = t(rp$cptable), splits = splits,
method = method, parms = init$parms, control = controls,
functions = functions)
}
if (ncat > 0)
ans$csplit <- catmat + 2L
if (model) {
ans$model <- m
if (missing(y))
y <- FALSE
}
if (y)
ans$y <- Y
if (x) {
ans$x <- X
ans$wt <- wt
}
ans$ordered <- isord
if (!is.null(attr(m, "na.action")))
ans$na.action <- attr(m, "na.action")
if (!is.null(xlevels))
attr(ans, "xlevels") <- xlevels
if (method == "class")
attr(ans, "ylevels") <- init$ylevels
class(ans) <- "rpart"
ans
}
<environment: namespace:rpart>
```

1.5.3 KS (KS.TEST)

```
function (x, y, ..., alternative = c("two.sided", "less", "greater"),
exact = NULL)
```

```
{
pkolmogorov1x <- function(x, n) {
if (x <= 0)
return(0)
if (x >= 1)
return(1)
j <- seq.int(from = 0, to = floor(n * (1 - x)))
1 - x * sum(exp(lchoose(n, j) + (n - j) * log(1 - x -
j/n) + (j - 1) * log(x + j/n)))
}
alternative <- match.arg(alternative)
DNAME <- deparse(substitute(x))
x <- x[!is.na(x)]
n <- length(x)
if (n < 1L)
stop("not enough 'x' data")
PVAL <- NULL
if (is.numeric(y)) {
DNAME <- paste(DNAME, "and", deparse(substitute(y)))
y <- y[!is.na(y)]
n.x <- as.double(n)
n.y <- length(y)
if (n.y < 1L)
stop("not enough 'y' data")
if (is.null(exact))
exact <- (n.x * n.y < 10000)
METHOD <- "Two-sample Kolmogorov-Smirnov test"
TIES <- FALSE
n <- n.x * n.y/(n.x + n.y)
w <- c(x, y)
z <- cumsum(ifelse(order(w) <= n.x, 1/n.x, -1/n.y))
if (length(unique(w)) < (n.x + n.y)) {
```

```
warning("cannot compute correct p-values with ties")
z <- z[c(which(diff(sort(w)) != 0), n.x + n.y)]
TIES <- TRUE
}
STATISTIC <- switch(alternative, two.sided = max(abs(z)),
greater = max(z), less = -min(z))
nm_alternative <- switch(alternative, two.sided = "two-sided",
less = "the CDF of x lies below that of y", greater = "the CDF of x lies
above that of y")
if (exact && (alternative == "two.sided") && !TIES)
PVAL <- 1 - .C("psmirnov2x", p = as.double(STATISTIC),
as.integer(n.x), as.integer(n.y), PACKAGE = "stats")$p
}
else {
if (is.character(y))
y <- get(y, mode = "function")
if (mode(y) != "function")
stop("'y' must be numeric or a string naming a valid function")
if (is.null(exact))
exact <- (n < 100)
METHOD <- "One-sample Kolmogorov-Smirnov test"
TIES <- FALSE
if (length(unique(x)) < n) {
warning("cannot compute correct p-values with ties")
TIES <- TRUE
}
x <- y(sort(x), ...) - (0:(n - 1))/n
STATISTIC <- switch(alternative, two.sided = max(c(x,
1/n - x)), greater = max(1/n - x), less = max(x))
if (exact && !TIES) {
PVAL <- if (alternative == "two.sided")
1 - .C("pkolmogorov2x", p = as.double(STATISTIC),
as.integer(n), PACKAGE = "stats")$p
else 1 - pkolmogorov1x(STATISTIC, n)
}
nm_alternative <- switch(alternative, two.sided = "two-sided",
less = "the CDF of x lies below the null hypothesis",
greater = "the CDF of x lies above the null hypothesis")
}
names(STATISTIC) <- switch(alternative, two.sided = "D",
greater = "D^+", less = "D^-")
pkstwo <- function(x, tol = 1e-06) {
if (is.numeric(x))
x <- as.vector(x)
else stop("argument 'x' must be numeric")
p <- rep(0, length(x))
p[is.na(x)] <- NA
IND <- which(!is.na(x) & (x > 0))
if (length(IND)) {
p[IND] <- .C("pkstwo", as.integer(length(x[IND])),
p = as.double(x[IND]), as.double(tol), PACKAGE = "stats")$p
}
return(p)
}
if (is.null(PVAL)) {
PVAL <- ifelse(alternative == "two.sided", 1 - pkstwo(sqrt(n) *
STATISTIC), exp(-2 * n * STATISTIC^2))
}
RVAL <- list(statistic = STATISTIC, p.value = PVAL, alternative =
nm_alternative,
method = METHOD, data.name = DNAME)
class(RVAL) <- "htest"
return(RVAL)
}
<environment: namespace:stats>
```

1.5.4 Area Under ROC (ROC.AREA)

```
function (obs, pred)
{
id <- is.finite(obs) & is.finite(pred)
obs <- obs[id]
pred <- pred[id]
n1 <- sum(obs)
n <- length(obs)
A.tilda <- (mean(rank(pred)[obs == 1]) - (n1 + 1)/2)/(n -
    n1)
stats <- wilcox.test(pred[obs == 1], pred[obs == 0], alternative =
"great")
return(list(A = A.tilda, n.total = n, n.events = n1, n.noevents = sum(obs
==
0), p.value = stats$p.value))
}
```

Appendix B

The JAVA Codes

```
package jibck;

import java.io.*;
import java.util.Arrays;
//import org.apache.commons.math.stat.inference.*;
//import jsc.independentsamples.SmirnovTest.*;
//import jsc.*;

//import java.util.*;
//import java.math.*;
/**
 *
 * @author Arthur
 */
public class Jibck {

 //Given an array data1[1..n1], and an array data2[1..n2], this routine returns the K-S
//statistic d, and the significance level prob for the null hypothesis that the data sets
// are drawn from the same distribution.
//Small values of prob show that the cumulative distribution
//function of data1 is significantly different from that of data2.
//The arrays data1 and data2 are modified by being sorted into ascending order.
   public static double[] somersdetc(double data1[], double data2[]) {
       int n1 = data1.length;
       int n2 = data2.length;
       int j1 = 1, j2 = 1;
       int counter1 = 1;
```

```
    double ks = 0, d1, d2, dt, en1, en2, en, en3, fn1 = 0.0, fn2 = 0.0, fn3 = 0.0, t1, c1,
somer, tau, goodman;
    double eps2 = .002;  //percent increase in cdf worth noting.
    double nc1, nd1, tied1;
    double nc2, nd2, tied2;
    double nc3, nd3, tied3;
    double prev1;
    /*
    double odds5=0,odds10=0,odds20=0,odds30=0;
    double podds5=0,podds10=0,podds20=0,podds30=0;
    double aodds5=0,aodds10=0,aodds20=0,aodds30=0;
    double apodds5=0,apodds10=0,apodds20=0,apodds30=0;
    double bodds5=0,bodds10=0,bodds20=0,bodds30=0;
    double bpodds5=0,bpodds10=0,bpodds20=0,bpodds30=0;
     */
    double[] oddscut = {0.05, 0.10, 0.20, 0.30};
    double[][] oddsinfo = new double[oddscut.length][2]; //the odds at the percent
    double[][] boddsinfo = new double[oddscut.length][2];//before the percent
    double[][] aoddsinfo = new double[oddscut.length][2];//after the percent
    double oddsme = 0.000001; //odds margin of error

    //if in the end it is negative one then something didnt workout, never within m.e.
    for (int i = 0; i < oddscut.length; i++) {
      oddsinfo[i][0] = -1;
      oddsinfo[i][1] = -1;
    }

    nc2 = 0;
    nd2 = 0;
    Arrays.sort(data1);
    Arrays.sort(data2);
    en1 = n1;
```

```
en2 = n2;
en3 = n1 + n2;
prev1 = 0.0;
nc1 = 0;
nd1 = 0;
tied1 = 0;
while (j1 < n1 && j2 < n2) {
   //advance data
   d1 = data1[j1];
   d2 = data2[j2];
   if (d1 <= d2 && j1 < (n1)) {//Next step is in data1.
      j1++;
   }
   if (d2 <= d1 && j2 < (n2)) {//Next step is in data2.
      j2++;
   }
   //basically advance j1 and j2 with these loops to the next yhat value
   //as many yhat values repeat due to the same x values

   //if (j1>99990) {System.out.println("h1 j1=" + j1);}
   if (j1 < n1) {
      while ((data1[j1] == data1[j1 - 1]) && (j1 < (n1 - 1))) {
         //if (j1>99990) {System.out.println("h2 j1=" + j1);}
         j1++;
      }
   }
   if (j2 < n2) {
      while ((data2[j2] == data2[j2 - 1]) && (j2 < (n2 - 1))) {
         j2++;
      }
   }
```

```
    fn3 = (double) (j1 + j2) / en3;
    for (int i = 0; i < oddscut.length; i++) {
       if (Math.abs(fn3 - oddscut[i]) < oddsme) {
          oddsinfo[i][0] = (double) (n2 - j2) / (double) (n1 - j1);
          oddsinfo[i][1] = fn3;
       }
    }

    //for calc KS
    fn1 = (double) j1 / en1;
    fn2 = (double) j2 / en2;
    if ((dt = Math.abs(fn2 - fn1)) > ks) {
       ks = dt;
    }
    //fn are cdfs fn3 is combined cdf at this point - j1 and j2
    fn3 = (double) (j1 + j2) / en3;
    if (fn3 > (prev1 + eps2)) {
    //if the combined cdf has increased by approx at least .2%
       prev1 = ((double) j1 + (double) j2) / en3;
    //keep track of previous combined cdf
       if (counter1 >= 1) {
          tied1 = tied1 + (j1 - nc2) * (j2 - nd2);
    //tied equals numbers within same combined cdf
          nd1 = nd1 + ((double) j1 - nc2) * ((double) n2 - (double) j2);
          nc2 = (double) j1;
          nd2 = (double) j2;
       }
       counter1 = counter1 + 1;
    }
}

//the next two while lopps are for when one dataset isnt done
```

```
//like either data1 or data2 is still not at the end
while (j1 < (n1)) {
   fn1 = (double) j1++ / en1;
   if ((((double) j1 + (double) j2) / en3) == 1) {
      if (counter1 >= 1) {
         tied1 = tied1 + ((double) j1 - nc2) * ((double) j2 - nd2);
         nd1 = nd1 + ((double) j1 - nc2) * ((double) n2 - (double) j2);
         nc2 = (double) j1;
         nd2 = (double) j2;
         //System.out.println("h3 j1=" + j1);
      }
      counter1 = counter1 + 1;
   }
}
while (j2 < (n2)) {  //If we are not done...
   fn2 = (double) j2++ / en2;
   if ((((double) j1 + (double) j2) / en3) == 1) {
      if (counter1 >= 1) {
         tied1 = tied1 + ((double) j1 - nc2) * ((double) j2 - nd2);
         nd1 = nd1 + ((double) j1 - nc2) * ((double) n2 - (double) j2);
         nc2 = (double) j1;
         nd2 = (double) j2;
         //System.out.println("h3 j2=" + j2);
      }
      counter1 = counter1 + 1;
   }
}
//for calc KS
fn1 = (double) j1 / en1;
fn2 = (double) j2 / en2;
if ((dt = Math.abs(fn2 - fn1)) > ks) {
   ks = dt;
}

fn3 = (double) (j1 + j2) / en3;
for (int i = 0; i < oddscut.length; i++) {
   if (Math.abs(fn3 - oddscut[i]) < oddsme) {
      oddsinfo[i][0] = (double) (n2 - j2) / (double) (n1 - j1);
      oddsinfo[i][1] = fn3;
   }
}

t1 = (double) n1 * (double) n2;
nc1 = t1 - nd1 - tied1;
/*
System.out.println("tied1= " + tied1);
System.out.println("nc1= " + nc1);
System.out.println("nd1= " + nd1);
System.out.println("t1= " + t1);
System.out.println("sum of parts= " + (tied1 + nc1 + nd1));
 */
c1 = (nc1 + .5 * (t1 - nc1 - nd1)) / t1;
somer = (nc1 - nd1) / t1;
somer = Math.abs(somer);
   // somer can be negative or positve but gini the positive of it.

goodman = (nc1 - nd1) / (nc1 + nd1);
tau = (nc1 - nd1) / (.5 * (en3 * (en3 - 1)));
double[] someretc1 = new double[20];
someretc1[0] = somer;
someretc1[1] = c1;
someretc1[2] = goodman;
someretc1[3] = tau;
```

```
        someretc1[4] = ks;  //want to calculate KS here to get better efficiency and not
use ks func

        int h = 5;
        for (int i = 0; i < oddscut.length; i++) {
            someretc1[h + i] = oddsinfo[i][0];
            h++;
            someretc1[h + i] = oddsinfo[i][1];
        }
        return someretc1;
    }

    public static double TwoSampKS(double[] data1, double[] data2) {
        //void kstwo(float data1[], unsigned long n1, float data2[], unsigned long n2,float
*d, float *prob)
        double theks = 0;
        int n1 = data1.length;
        int n2 = data2.length;
        Arrays.sort(data1);
        Arrays.sort(data2);
        int en1 = n1;
        int en2 = n2;
        double d = 0.0, d1 = 0, d2 = 0, fn1 = 0, fn2 = 0, dt = 0;
        int j1 = 0, j2 = 0;
        while (j1 < n1 && j2 < n2) {
            d1 = data1[j1];
            d2 = data2[j2];
            if (d1 <= d2) {
                j1++;
                fn1 = (double) j1 / (double) en1;
            }
            if (d2 <= d1) {
```

```
            j2++;
            fn2 = (double) j2 / (double) en2;
         }
         dt = Math.abs(fn2 - fn1);
         if (dt > d) {
            d = dt;
         }
      }
//en=sqrt(en1*en2/(en1+en2));
//*prob=probks((en+0.12+0.11/en)*(*d)); Compute significance.

      theks = d;
      return theks;
   }

   public static void writeresults(String thedir, String fname, double thestats[][]) {
      try {
         PrintStream writer = new PrintStream(thedir + fname);
         writer.print("Abs(SomersD),c-statistic,Goodman,Tau-alpha,KS-statistic,");
         writer.print("odds, percent, odds, percent, odds, percent, odds, percent, \n");
         for (int i = 0; i < thestats.length; i++) {
            for (int j = 0; j < thestats[i].length; j++) {
               writer.print(thestats[i][j]);
               if (j < thestats[j].length - 1) {
                  writer.print(',');
               } else {
                  writer.print('\n');
               }
            }
         }
      } catch (IOException e) {
      }
   }
```

```
public static double[][] readdata(String fname, int nc, int nr, int sc) {
   //sc is column to start with
   double[][] thedata = new double[nr][nc];
   BufferedReader br = null;

   try {

      int h, i;
      br = new BufferedReader(new FileReader(fname));
      String line = null;
      h = 0;
      i = 0;
      line = br.readLine();
      System.out.println(line);
      while ((line = br.readLine()) != null) {
         String[] values = line.split(",");
         //Do necessary work with the values, here we just print them out
         for (String str : values) {
            if (i > (sc - 1)) {
               thedata[h][(i - sc)] = Double.parseDouble(str);
            }
            i++;
            //System.out.println(str);
         }
         i = 0;
         h++;
         //System.out.println();
      }
   } catch (FileNotFoundException ex) {
   } catch (IOException ex) {
   } finally {
      try {
```

```
                if (br != null) {
                    br.close();
                }
            } catch (IOException ex) {
            }

        }

        return thedata;
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        String thedir = "C:/Documents and Settings/user/Desktop/Jib/";
        String fn = thedir + "90L_Apop.csv";
        int nrp = 1000000;
        int nc = 11;
        double[][] popdata1 = new double[nrp][nc];
        int sc = 0;
        popdata1 = readdata(fn, nc, nrp, sc);
        System.out.println("read data function done");
        for (int j = 0; j < 10; j++) {
            for (int k = 0; k < 11; k++) {
                System.out.print(" " + popdata1[j][k] + " ");
            }
            System.out.println();
        }
        System.out.println("done with loop");
        int nr = 1000;
```

```
nc = 11;
double[][] LRbetas = new double[nr][nc];
double[][] LGbetas = new double[nr][nc];
fn = thedir + "coefLR.csv";
sc = 1;
LRbetas = readdata(fn, nc, nr, sc);
fn = thedir + "coefLG.csv";
sc = 1;
LGbetas = readdata(fn, nc, nr, sc);
for (int j = 0; j < 10; j++) {
   for (int k = 0; k < 11; k++) {
      System.out.print(" " + LGbetas[j][k] + " ");
   }
   System.out.println();
}
int nrp0 = 100000;
int nrp1 = 900000;
int iters = 1000;
//iters = 50;
double LRtemphat = 0;
double LGtemphat = 0;
int r0 = 0;
int r1 = 0;
double[] LRpopyhats0 = new double[nrp0];
double[] LRpopyhats1 = new double[nrp1];
double[] LGpopyhats0 = new double[nrp0];
double[] LGpopyhats1 = new double[nrp1];

//double[] theks = new double[iters];
double[] tempstats = new double[20];
//only 14
double[][] LGthestats = new double[iters][16];
double[][] LRthestats = new double[iters][16];
```

```
//calculating yhats for logistic regression
for (int m0 = 0; m0 < iters; m0++) {
   r0 = 0;
   r1 = 0;
   for (int j = 0; j < nrp; j++) {
      LGtemphat = LGbetas[m0][0];
      LRtemphat = LRbetas[m0][0];
      for (int p = 1; p < nc; p++) {
         LGtemphat = LGtemphat + LGbetas[m0][p] * popdata1[j][p];
         LRtemphat = LRtemphat + LRbetas[m0][p] * popdata1[j][p];
      }
      if (j<100){System.out.println("LGtemphat="+LGtemphat);}
      if (popdata1[j][0] < .1) {
         //popyhats00[r0][m0] = Math.exp((LGtemphat / (LGtemphat + 1)));
         LGpopyhats0[r0] = (1.0 / (1.0 + Math.exp(-1.0 * LGtemphat)));
         //LGpopyhats0[r0] = LGtemphat;
         LRpopyhats0[r0] = LRtemphat;
         r0++;
      } else {
         //popyhats01[r1][m0] = Math.exp((LGtemphat / (LGtemphat + 1)));
         LGpopyhats1[r1] = (1.0 / (1.0 + Math.exp(-1.0 * LGtemphat)));
         //LGpopyhats1[r1] = LGtemphat;
         LRpopyhats1[r1] = LRtemphat;
         r1++;
      }
   }

   //first logistic regression
   tempstats = somersdetc(LGpopyhats0, LGpopyhats1);
   //System.out.println("The gini equals = " + tempstats[0]);
   //System.out.println("The KS equals from somers function = " + tempstats[4]);
   //theks[m0] = TwoSampKS(LGpopyhats0, LGpopyhats1);
   //System.out.println("The KS equals = " + theks[m0]);
```

```
      LGthestats[m0][0] = tempstats[0];
      LGthestats[m0][1] = tempstats[1];
      LGthestats[m0][2] = tempstats[2];
      LGthestats[m0][3] = tempstats[3];
      LGthestats[m0][4] = tempstats[4];  //not sure which func for KS to use
      //LGthestats[m0][4] = theks[m0];  //same results
      for (int h = 0; h < 10; h++) {
        //System.out.println("odds info stuff = " + tempstats[5 + h]);
        LGthestats[m0][5 + h] = tempstats[5 + h];
      }

      //second linear regression
      tempstats = somersdetc(LRpopyhats0, LRpopyhats1);
      //System.out.println("The gini equals = " + tempstats[0]);
      //System.out.println("The KS equals from somers function = " + tempstats[4]);
      //theks[m0] = TwoSampKS(LRpopyhats0, LRpopyhats1);
      //System.out.println("The KS equals = " + theks[m0]);
      LRthestats[m0][0] = tempstats[0];
      LRthestats[m0][1] = tempstats[1];
      LRthestats[m0][2] = tempstats[2];
      LRthestats[m0][3] = tempstats[3];
      LRthestats[m0][4] = tempstats[4];  //not sure which func for KS to use
      //LRthestats[m0][4] = theks[m0];   //same results
      for (int h = 0; h < 10; h++) {
        //System.out.println("odds info stuff = " + tempstats[5 + h]);
        LRthestats[m0][5 + h] = tempstats[5 + h];
      }
    }
    writeresults(thedir, "LGresults.csv", LGthestats);
    writeresults(thedir, "LRresults.csv", LRthestats);
  }
}
```

BIOGRAPHY

NAME	Vesarach Aumeboonsuke
ACADEMIC BACKGROUND	BBA, Finance Major, Assumption University, Thailand MSc Finance & Investment, Brunel University, UK
PRESENT POSITION	Lecturer, Finance Department, Martin De Tour School of Management & Economics, Assumption University
EXPERIENCES	Lecturer, Finance Department, Martin De Tour School of Management & Economics, Assumption University