THE INTERACTION OF SAMPLING RATIO AND MODELLING METHOD IN PREDICTION OF BINARY TARGET WITH RARE TARGET CLASS

Steven Hirschowitz

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

Johannesburg, November 2008

Declaration

I declare that this research report is my own, unaided work. It is being submitted for the degree of Master of Science in the University of Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other university.

Signed on the 21st November 2008

Steven Hirschowitz

Abstract

In many practical predictive data mining problems with a binary target, one of the target classes is rare. In such a situation it is common practice to decrease the ratio of common to rare class cases in the training set by under-sampling the common class. The relationship between the ratio of common to rare class cases in the training set and model performance was investigated empirically on three artificial and three real-world data sets. The results indicated that a flexible modelling method without regularisation benefits in both mean and variance of performance from a larger ratio when evaluated on a criterion sensitive to overfitting, and benefits in mean but not variance of performance when evaluated on a criterion less sensitive to overfitting. For an inflexible modelling method and a flexible method with regularisation, the effects of a larger ratio were less consistent. In no circumstances, however, was a larger ratio found to be detrimental to model performance, however measured.

Table of Contents

List of Figures

List of Tables

1 Introduction

In many practical predictive data mining problems with a binary target, one of the classes is rare. For example in prediction of take-up of a new home loan in the next three months, the 'positive' outcome may represent only about 2% of the population, while in fraud detection it is likely to represent much fewer than 1% of transactions (Brause, Langsdorf and Hepp , 1999; Hassibi, 2000). Usually, however, the rare class is that of most interest, and it may be far more costly to miss an instance of the rare class than to label an instance of the common class as belonging to the rare class.

Over the past 10-15 years, this issue has assumed increasing prominence in the machine learning and data mining literature (Chawla, Japkowicz and Kolcz, 2004). It has come to be recognised that rare class applications are far from uncommon in practice, and prediction of rare classes has relevance in a wide variety of disciplines, for example learning word pronunciations (van den Bosch *et al*, 1997), predicting pre-term births (Grzymala-Busse *et al*, 2000), predicting telecommunication equipment failures (Weiss and Hirsh, 1998) and detecting oil spills from satellite images (Kubat, Holte and Matwin, 1998).

It has become increasingly recognised that a rare target class presents challenges with regard to both model fitting and model evaluation. With regard to fitting, it has been claimed that “standard classifiers tend to be overwhelmed by the large classes and ignore the small ones” (Chawla, Japkowicz and Kolcz, 2004). With regard to model evaluation, classification accuracy, the most commonly used measure in the machine learning literature, may be inappropriate. If the data are noisy, as is common in data mining problems, there may be no region of the input space in which the rare class is more probable than the common class. In such a situation the Bayes-optimal classifier, assuming equal misclassification costs (amongst other scenarios), classifies all cases into the common class. Hence measures need to be found that give sufficient weight to the ability of the model to identify regions of the input space in which the rare class is more probable or less probable than the population prior.

For the reasons given above, as well as to limit the data extraction and processing time, it is common data mining practice to create a training set containing an equal number of rare and common class cases, usually (after removing a holdout set for evaluation purposes) by including all of the available rare class cases and a random sample of equal size of the common class cases, fitting the model, and then adjusting the predicted probabilities based on the population priors (Georges, 2004). In other words, if N_0 is the number of common class cases and N_1 is the number of rare class cases in the training set, under-sampling of common class cases is used to produce a training set with a $N_0 : N_1$ ratio of 1:1. The above-mentioned practice is known in various disciplines as sampling stratified on the target, separate sampling, case-control sampling, choice-based sampling, outcome-dependent sampling or under-sampling. In this report, the $N_0 : N_1$ ratio in the training set will usually be referred to as the 'sampling ratio' and the ratio x:1 will usually be abbreviated simply as ratio x.

This practice, however, is wasteful of the information available from the large number of cases of the common class (Chan and Stolfo, 1998). This would seem to be particularly relevant if the dimensionality is high — large number of predictors — and hence the data are sparse in the input space, even for what might seem like a large number of cases. It seems reasonable to ask the question: for flexible modelling methods, does using the entire available data set, or at least a larger sampling ratio, help to reduce the variance (tendency to model the noise in the data) by weighting down random rare class outcomes with a greater number of common class outcomes with similar values on the predictors, when compared to equal sampling? Figure 1 gives an illustration of why this question is pertinent.

This practice also gives rise to theoretical problems. In the case of maximum likelihood estimation, the likelihood function changes under separate sampling (Manski and Lerman, 1977). In practice, however, this complication is ignored. When a multiplicative intercept modelling method, such as logistic regression or multilayer perceptron with logistic output activation function, is used, there is theoretical work (Hsieh, Manski and McFadden, 1985; Weinberg and Wacholder, 1993) showing that the effect is only to bias the intercept term, and this bias is correctable. This does not, however, hold for other model forms, and in general all model parameters may be biased by optimisation of a likelihood that ignores sampling methodology (Manski and Lerman, 1977).

Over the past decade, numerous empirical studies have been published in the machine learning and artificial intelligence literature, comparing various sampling strategies using specified model forms and evaluation criteria. However in general, as discussed in detail in the literature review, these studies have been limited in several ways. They have generally compared only equal-size sampling to training using all available data, rather than considering a range of sampling ratios. Very few have looked at the variability of model performance due to sampling. Very little, if any, published research has considered the interaction between a range of sampling ratios and model type on the variability of model performance.

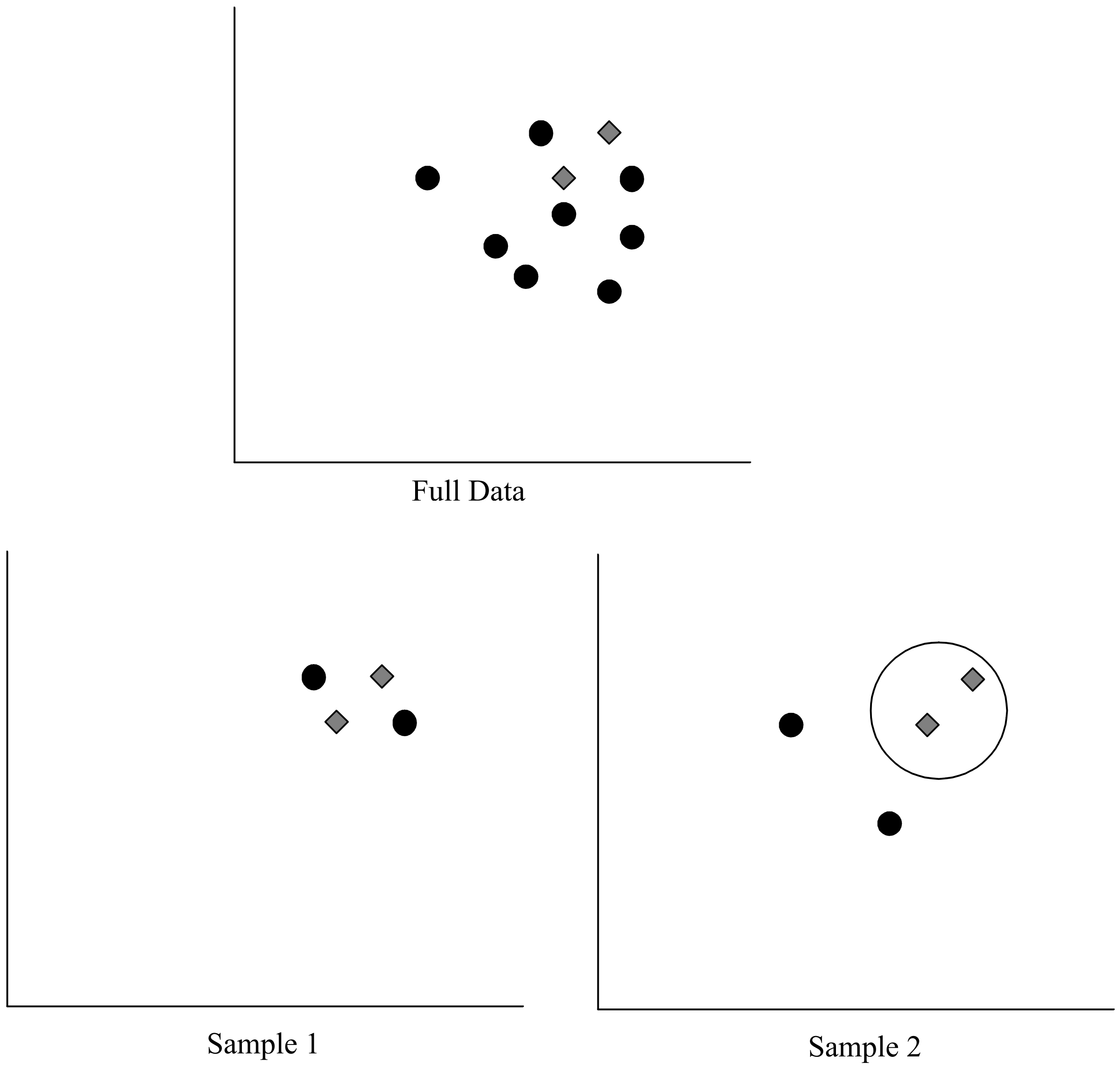


Figure 1.1: Illustrative example in 2 dimensions of a complete data set and two possible training samples generated by under-sampling the common class to produce a sampling ratio of 1:1. The diamonds represent the rare class cases (included in both training samples) and the dots represent the common class cases under-sampled using random sampling. With a flexible modelling method, a model trained on sample 2 'finds' a high probability region (circled) while a model trained on sample 1 may not.

This report is organised as follows. Chapter 2 provides a brief review of relevant literature and theory relating to sampling methods, modelling methods, evaluation methods and empirical studies that have been conducted. Some gaps in the literature that are addressed in this study are identified. Chapter 3 outlines the hypotheses tested and the methodology and experimental design used to test them empirically. Chapter 4 describes the results of the experiment on three artificial data sets and Chapter 5 describes the results on three real-world data sets. Chapter 6 summarises the conclusions and suggests possible directions for future research. Appendix A provides details of the three real-world data sets used in the study. Appendix B gives a listing of the SAS code used to implement the experiment.

2 Literature and Theory Review

Note that in this report, rare class cases will also be referred to as ‘positives’ or ‘1s’, while common class cases will also be referred to as ‘negatives’ or ‘0s’. Observations are also referred to as ‘examples’ or ‘cases’. Also, data sets are frequently referred to as ‘domains’ in the machine learning literature, even though the term more strictly refers to the population from which the observed data have arisen. Predictors are alternatively referred to as ‘inputs’. Model parameters are alternatively referred to as ‘weights’.

The review covers the following broad topics: sampling methods, modelling methods, evaluation methods, comparative empirical studies and gaps identified in the literature.

2.1 Sampling Methods

There is much statistical literature (e.g. Cochran, 1977; Chaudhuri and Stenger, 2005) concerning sampling methodology for designed experiments or surveys, where the aim is to design the selection of sampling units in an optimal way for a given purpose, such as to maximise precision of estimation of a population parameter or parameters of interest for fixed total sample size or cost, before carrying out the experiment or survey. Stratified sampling is frequently employed so that sufficient precision can be obtained for population parameters of interest within each of the classes (strata), and indeed an optimal allocation for estimation of an overall population parameter of interest can be defined. For most purposes, a rare class would be over-represented in the sample design.

In the case of data mining, however, we are faced with a different situation. The data have not been collected for a specific modelling or estimation purpose, but are simply ‘there’ because operational systems have collected it (Potts, 1998). Despite this, we must assume that the available data are representative of the population to which we wish to generalise – which frequently includes cases which will arise in the future – and that we can draw representative sub-samples from it (Fayyad, 1996; Mahadevan *et al.*, 2002). Also, the data size is frequently large. For example, in the financial services sector, in prediction of sale of new home loan, it is not uncommon to have several hundred thousand cases (customers) on which to build a predictive model, and in the case of fraud detection, there may be hundreds of millions of cases (transactions) monthly. Yet, in the home loan example, perhaps only 2% of cases have the outcome ‘sale’, and in the fraud detection example, far less than 1% of cases have the outcome ‘fraud’ (Brause, Langsdorf and Hepp , 1999; Hassibi, 2000). Hence the problem becomes one not of selecting sampling units optimally for fixed sample size, but rather whether some manipulation of cases on which data have already been collected can improve model performance, or shorten training time without degrading performance. In particular, can we reduce the overall sample size by under-sampling the common class cases without significantly increasing in the variability of performance or worsening the mean of performance?

The most common strategy adopted in practice in the data mining community (see e.g. Georges, 2004) is as follows:

a) split off a test set from the available data set.
b) put all of the remaining positive cases into the training set $T*$ say.
c) under-sample the negative cases i.e. take a simple random sample of negative cases of equal size to the existing $T*$ and put these in $T*$.
d) fit the model on $T*$.
e) when scoring unseen data, adjust the predicted probabilities based on the population priors so as to reflect population estimates.

In this study I will refer to the data set *after* removing a test sample as the ‘full data set’ and the actual data set used for model estimation and selection, after any sampling has been done, as the ‘training’ set. I will also refer to the $N_0 : N_1$ ratio in the training set as the ‘sampling ratio’.

Intuitive arguments have been made for and against the above strategy. On the one hand, this strategy is wasteful of the information available from the large number of cases with negative outcomes (Chan and Stolfo, 1998). This would seem particularly important if, as is typical in the data mining setting, a large number of inputs is available for model inclusion, since removal of negatives could potentially induce false separation of the classes. “A densely populated input space is required to fit highly complex models. In assessing how much data is available for data mining, the dimension of the problem must be considered” (Potts, 2000). Further, for many model types it introduces an element of randomness into a process that, apart from training / validation / test data partition, is deterministic (Drummond and Holte, 2003). On the other hand, it is possible that if the full training set is used, model fitting could be hampered by the large number of negative cases ‘swamping’ the positive cases (Chawla, Japkowicz and Kolcz, 2004). Perhaps there is some validity in both of these arguments, and it is optimal to take some other sampling ratio? And perhaps this optimal ratio depends on the modelling method used?

It should be noted that various alternative sampling strategies have been suggested in the machine learning literature. These strategies, though interesting, were not tested in this study. Some have, however, proved useful in some empirical studies, and they are referred to in Section 2.4. They are listed below as they could provide a basis for future research. A summary of some of these strategies, along with algorithms for implementing them, is given in Zhou and Liu, 2006. These strategies are:

- ‘Over-sampling’ resamples the positive cases with replacement, to create a larger positive sample equal in size or equal in total misclassification cost (if a suitable cost matrix can be defined) to the existing negative sample.

- Synthetic Minority Over-sampling Technique, or SMOTE, is a deterministic over-sampling technique for artificially increasing the size of the positive class suggested by Chawla *et al* (2002). Let *k* be the ‘amplification’ of the minority class required to match the majority class either in size or in total misclassification cost. Then for each positive example, the *k* nearest neighbours (using a suitable metric) of positive class are found. One positive example is then created at the midpoint of the line segment joining this case with each of its *k* nearest neighbours. Variations of SMOTE have been proposed (Han, Wang and Mao, 2005).

- Nickerson , Japkowicz and Millos (2001) suggest performing separate cluster analyses on the inputs for the positive and negative cases, and then randomly over-sampling from each resulting cluster so as to make all clusters for each class equal in size, and the total number of positives and negatives in the training data equal.

- ‘Simple under-sampling’ selects a simple random sub-sample of the negative cases equal in size or in total misclassification cost to the positive cases. When equality is in size, this is the common strategy discussed above.

- ‘Focused’ or ‘intelligent’ under-sampling also creates a sub-sample of negatives equal in size or in total misclassification cost to the positives. However, it makes an attempt to select these examples so as to remove redundant and / or borderline cases. All of the positive examples and half of the negative examples at random are put in the training set T^*. In the redundancy step, each of the remaining negative examples in turn is then considered for inclusion in T^*: if its 1-nearest neighbour in T^* is also negative it is regarded as redundant and not included, otherwise it is included. In the borderline step, an attempt is made to remove examples close to the class boundaries - "they are unreliable because even a small amount of attribute noise can send the example to the wrong side of the boundary" (Zhou and Liu, 2006). A negative case is selected at random from T^*: if its 1-nearest neighbour in T^* is positive then it is considered borderline and removed. This process is continued until all of the negative examples have been examined. Finally negative examples are removed at random until T^* contains the required numbers of positive and negative examples. A number of variations on this algorithm have been suggested (Kubat and Matwin, 1997; Laurikkala, 2001; Barandela *et al*, 2004; Stefanowski and Wilk, 2006).

- Ensemble models combine models built using several of the sampling methods listed above. Chan and Stolfo (1998) suggest partitioning the negatives into (N_1 / N_0) subsets. Each of these subsets in turn is combined with all the positives and a model fitted. Then an ensemble method (such as averaging posterior probabilities) is used to get the final estimate. This procedure has the advantage that it doesn't waste any information, but statisticians intuitively feel more comfortable with simultaneous optimisation strategies (Potts, 1998). Estabrooks, Jo and Japkowicz (2004) suggest combining models built on a number of training sets, each employing a different rate of under- and over-sampling.

2.2 Modelling Methods and Model Estimation

Two types of parametric or 'global' models were considered - logistic regression and multilayer perceptron neural network. These two model types were chosen to represent a more rigid method and a more flexible approach. Since the intention of this study was not to concentrate on the methods themselves, but rather how they interact with sampling ratio, they will only be sketched briefly.

2.2.1 Logistic Regression

A logistic regression model with p inputs $X_1,\dots,X_p$ has the form:

$$\log(E(Y)/(1-E(Y)) = w_0 + w_1X_1 + w_2X_2 + \dots + w_pX_p;$$

$$Y \sim Bernoulli(E(Y))$$

where $w_i, i = 0,\dots,p$ are the weights (parameters) to be estimated in the fitting process. Since the relationship between the inputs and the output is constrained to be linear, logistic regression was used as the 'inflexible' model form in this study.

2.2.2 Multilayer Perceptron

The multilayer perceptron (MLP) was used as the 'flexible' model form in this study. A multilayer perceptron is a type of feedforward neural network model. It fits a flexible, nonlinear function of the inputs (Potts, 2000). A MLP with a single hidden layer and sufficient hidden neurons is a 'universal approximator', i.e. can approximate any continuous function to any degree of accuracy (Ripley, 1996; Potts, 2000).

A MLP with p inputs and one hidden layer with h neurons has the functional form:

$$g_0^{-1}(E(Y)) = w_0 + w_1H_1 + \cdots + w_hH_h$$

where

$$H_i = g_1(w_{0i} + w_{1i}X_1 + w_{2i}X_2 + \cdots + w_{pi}X_p), i = 1,\dots,h$$

and $w_0,\dots,w_h$ and $w_{01},\dots,w_{p1},w_{02},\dots,w_{p2},\dots,w_{0h},\dots,w_{ph}$ are the weights (parameters) to be estimated in the fitting process, and hence has $h(p+2)+1$ parameters.

The following choices must be made by the modeller:

- Number of hidden layers and number of hidden units.
- Method for standardising the inputs.
- Error distribution for the target.
- Objective function to be optimised.

- Hidden layer activation function g_1.
- Output activation function g_0.
- Iterative fitting method.
- Regularisation method (weight decay, early stopping, both or neither).
- Initialisation method.
- Convergence criteria.

Note that a MLP with no standardisation, no hidden layers and logistic output activation function is equivalent to logistic regression (Potts, 2000).

2.2.3 Maximum Likelihood Estimation

Here we are concerned with fitting of parametric or 'global' models, in particular logistic regression and multi-layer perceptron neural networks, as opposed to local models such as trees and kernel regression. For a parametric model, the parameters (weights) need to be estimated from the training data. Maximum likelihood estimation is by far the most common estimation method used in practice, as it has many desirable properties such as asymptotic efficiency and consistency.

Given a probability distribution for the target conditional on the predictors and (unknown) parameters, the likelihood function $L(\mathbf{w};\mathbf{y},\mathbf{X})$, where $\mathbf{w}$ is the vector of parameters, $\mathbf{y}$ is the vector of target outcomes and $\mathbf{X}$ is the predictor matrix, is the joint probability of the observed data, expressed as a function of the parameters (Potts, 2000). Maximum likelihood estimation finds those values of the parameters that maximise the likelihood function L (or, equivalently, minimise $-2\log(L)$).

It is important to note, however, that the likelihood under separate sampling is not the same as the likelihood under simple random sampling (see e.g. Manski and Lerman, 1977). In practice we maximise the likelihood under simple random sampling on the training set even if separate sampling was used. Manski and Lerman (1977) have shown that, in general, this leads to inconsistent estimates for the parameters.

In the special case of multiplicative intercept models (Hseih, Manski and McFadden, 1985), however, Weinberg and Wacholder (1993) have shown that the maximum likelihood parameter estimates (MLEs) under separate sampling are, except for the intercept, identical to those under simple random sampling, the desirable properties of MLEs apply to our estimates and likelihood ratio testing is valid. A multiplicative intercept model has the form, in the case of a binary target, of:

$$P(Y=1\,|\,\mathbf{x})/\{1-P(Y=1\,|\,\mathbf{x})\} = hf(\mathbf{x},\beta),$$

where f is, for fixed $\mathbf{x}$, a twice-differentiable function from $\Re^{p}$ to $\Re^{+}$, h is an unknown positive scalar, β is a p-vector of unknown parameters, and p is the number of parameters excluding the intercept. The intercept under separate sampling is biased. However, this bias can be corrected by subtracting $\ln(\frac{\pi_0 \rho_1}{\pi_1 \rho_0})$ from the estimate under simple random sampling, where:

π_0 and π_1	are the population priors for class 0 and 1 respectively, and
ρ_0 and ρ_1	are the overall proportions of class 0 and 1 in the training data.

The class of multiplicative intercept models includes both logistic regression and MLP with logistic output activation function (King and Zeng, 2004). Thus for all models used in this study, we can obtain valid maximum likelihood estimates, and the properties of MLEs hold, regardless of sampling ratio. However, the properties of MLEs are asymptotic in nature, and in practice we have a finite sample; in particular we may only have a small number of rare class cases, relative to the input space. Hence the need for empirical studies such as this.

2.3 Evaluation methods

2.3.1 Evaluation Data

In the classical statistical literature, models are traditionally evaluated on the same data set on which they are fitted. This is because of both the relatively small samples sizes usually available from designed experiments and the relative lack of flexibility of the model forms used, which inherently limits the ability to overfit. Regularisation techniques are also frequently adopted to try and prevent the model from overfitting, either by penalising an excessive number of model parameters (AIC, SBC) or excessive size of parameters (ridge regression). Numerous alternative penalty functions have been suggested, and can be justified as selecting models that are either asymptotically efficient or consistent for the 'true' model

(McQuarrie and Tsai, 1998). Harrell (2001) argues that, despite these penalties, traditional statistical modelling techniques *are* prone to some degree of overfitting. For example, in forward stepwise regression, tests used to determine whether to include an additional term in the model are based on distribution theory that assumes that the models compared were pre-specified, whereas, especially with a large number of predictors to potentially be included at each step, this is far from the case. Ye (1998) develops the concept of generalised degrees of freedom as a tool for properly comparing models of differing types and complexities. However, this will not be explored further here.

More recently, increased computing power has led statisticians to adopt resampling techniques for model evaluation. For example, in k-fold cross-validation (Stone, 1974; Geisser, 1975) the data set is partitioned into *k* subsets of approximately equal size and each subset is used exactly once as a test set, with the other (*k*-1) subsets combined being used to estimate and select the model. Cross-validation is also commonly used in the machine-learning literature, particularly when the data set is relatively small.

In the data mining context, we are frequently fortunate enough to have enough data to do split-sample evaluation. A subset of the data is used strictly as a test set for unbiased evaluation of the final model. The rest of the data are used for training, and this can be split into a 'training' data set used for fitting models and a 'validation' set used to choose among candidate models. In a data-rich situation, this is the best approach for both model selection and model assessment (Hastie, Tibshirani and Friedman, 2001). This is the approach that was used in this study.

2.3.2 Evaluation Criterion

Finding an appropriate evaluation criterion when the target class distribution is heavily imbalanced is a difficult task (Elazmeh, Japkowicz and Matwin, 2006) and the suitability of various criteria is still the subject of much debate. Hand (2005) argues, in the credit scoring setting, that only the sign and not the size of the difference between the score and the threshold matters, and criteria that depend on the size of this difference can be seriously misleading. Many other researchers (e.g. Ling, Huang and Zhang, 2003) take a different view. Moreover, several empirical studies (Van Hulse, Khoshgoftaar and Napolitano, 2007; Cui *et al*, 2008) have found that different conclusions can be reached depending on the evaluation criterion used.

Much of the machine-learning literature uses accuracy as the evaluation criterion. The fitted model assigns a class to each case in the test set, either through maximum posterior probability or ‘voting’. Accuracy is defined as the correct classification rate using a 50% threshold. However, numerous authors have pointed out that accuracy is not a good criterion for classification with rare target classes. Statistical decision theory states that each case should be assigned to the class giving minimum posterior expected cost (or, equivalently, maximum posterior expected profit). In this context, the accuracy criterion is only optimal under the assumption that the cost of falsely classifying a negative as a positive (false positive or FP) is equal to the cost of falsely classifying a positive as a negative (false negative or FN) (Provost, Fawcett and Kohavi, 1998). In most practical situations with rare outcomes, however, the cost of a FN (i.e. missing a positive) is much higher than the cost of a FP. As Georges (2004) and Drummond and Holte (2005) point out, if the relative costs of FP to FN are very far from the population ratio of positive to negative, one could question whether modelling is worthwhile at all, as the extreme decisions (classify all cases as positive or all as negative) could well be close to optimal.

If costs and population priors are known and classification decisions are made so as to minimise expected cost for each case, then models can be compared based on total misclassification cost on the test data. A cost matrix defines the relative cost of FP and FN. It is becoming popular in the machine learning community to compare models based on various assumed cost matrices, and even to make adjustments to the algorithms themselves based on relative misclassification costs of the classes – a field of research known as ‘cost-sensitive learning’.

In many practical situations, however, relative costs cannot accurately be defined. Further, Provost and Fawcett (2001) point out that relative costs can change over the time of model deployment. Hence it is desirable to have a measure that is invariant under differing cost assumptions. Moreover, for many applications we want measures of the accuracy of the posterior probabilities themselves, not just the value of the classification decisions based on these probabilities. As a first step, we may consider measures of the quality of ranking (Ling, Huang and Zhang, 2003).

One such measure is the area under the Receiver Operating Characteristic (ROC) curve (AUC), and for this reason this has become a standard measure of model performance in the machine learning community (Chawla, Japkowicz and Kolcz, 2004). The ROC curve is a plot of TP rate (‘specificity’) on the y-axis against FP rate (1-‘sensitivity’) on the x-axis. Because sensitivity and specificity are computed separately within each target level (Georges, 2001), ROC curves describe the predictive behaviour of a classifier independent of class distributions or error costs, so they decouple classification performance from these factors (Provost, Fawcett and Kohavi, 1998).

A point on the ROC curve represents a classifier. The (0,0) point corresponds to classifying all cases as negative, and the (1,1) point all cases as positive. The 'ideal' model is given by (0,1) – 100% TP rate and 0% FP rate. A 'model' that assigns classes at random achieves an AUC of 0.5. A model which generates a posterior probability of class membership (as opposed to just a 0/1 classification) generates a continuous set of points (curve) in the (x,y) space, with each point representing a specific decision threshold (cutpoint). According to Provost and Fawcett (2001),

> In situations where either the target cost distribution or class distribution is completely unknown, some researchers advocate choosing the classifier that maximises a single-number metric representing the average performance over the entire curve. A common whole-curve metric is "AUC", the Area Under the ROC Curve (Bradley, 1997). The AUC is equivalent to the probability that a randomly chosen positive instance will be rated higher than a negative instance, and thereby is also estimated by the Wilcoxon test of ranks (Hanley and McNeil, 1982).

The equivalence of AUC and Wilcoxon test of ranks means that the theory associated with the Wilcoxon test can be used to construct standard errors and hence also confidence intervals for AUC (Hanley and McNeil, 1982). This can be useful for determining if there is a statistically significant difference between the AUC generated by two different models.

A criticism of AUC is that for specific target conditions the classifier with the maximum AUC may be suboptimal in terms of total misclassification cost. Indeed, this criticism may be made of any single-number metric (Provost, Fawcett and Kohavi, 1998; Provost and Fawcett, 2001).

Another problem with AUC is that in many practical situations it is not just the ranking of the predicted probabilities of interest, but the actual values of the probabilities themselves. For example, in a database marketing context, we may develop several *separate* models on a customer base, each for propensity to buy a given product, and we may wish to market to each customer that product for which he or she has the highest posterior probability of purchase. Hence measures based on the difference between outcome and predicted probability are of interest.

Two such measures are log-likelihood loss and squared error loss (Friedman, 1997; Hastie, Tibshirani and Friedman, 2001). For log-likelihood loss, also referred to as cross-entropy loss or deviance, the loss – also known as squared deviance residual – for a case belonging to class G is $-2\log\widehat{p}_G(\mathbf{x})$, where $\widehat{p}_G(\mathbf{x})$ is the posterior (predicted) probability of belonging to class G. Squared error loss is given by $(1-\widehat{p}_G(\mathbf{x}))^2$.

2.4 Empirical Comparisons of Sampling Strategies using various Modelling Methods and Evaluation Criteria

Numerous empirical comparisons of sampling strategies have been published in the machine learning and artificial intelligence literature. These studies have compared sampling to create a ratio of (usually) 1:1 by either over-sampling the rare class, under-sampling the common class, creating synthetic examples or some combination of these three strategies with using the full data set. A recent trend is to compare also adaptation of the actual fitting algorithm to take account of relative misclassification costs ('cost-sensitive learning'). The majority of the studies have been done using Quinlan's C4.5 or C5.0 tree algorithm, with or without boosting and / or pruning. Relatively few have considered neural networks, regression, support vector machines, naïve Bayes, k-nearest neighbour, genetic and other algorithms.

It should be noted that trees seem to be the tool of choice among the machine learning community (Drummond and Holte, 2003, state, rather contentiously, that C4.5 "has become a de facto standard against which every new algorithm is judged"). However, trees have very different model structure to the models considered in this study (logistic regression and neural networks) in that, while they can be represented in the form of a statistical model, the models parameters are not fitted simultaneously to minimise an appropriate error function over the entire training data set. Rather, they are fitted in a stagewise, greedy fashion, without consideration as to what the 'global' optimum model will eventually look like. With unbalanced data, trees present their own set of problems: for example over-sampling tends to reduce the amount of pruning that occurs, while under-sampling often renders pruning unnecessary (Drummond and Holte, 2003). Hence there is a large amount of literature comparing pruning against not pruning under various sampling strategies.

The results of the studies are generally inconclusive. Part of the problem seems to be a lack of consistent methodology. Firstly, many different domains (data sets) are used, mostly real –

usually taken from the UCI repository (Asuncion and Newman, 2007), plus some other sources listed as mentioned – but occasionally also artificial. Secondly, among measures of 'best performance', different researchers have used accuracy, area under ROC curve, minimum cost under various cost assumptions or other criteria. Thirdly, among data partitioning methodologies some researchers have used a single test data set whilst others have used cross-validation techniques.

Of the studies specifically using neural networks, the results are not all consistent, though the majority find under-sampling the common class to be worse than either over-sampling the rare class or using the full data set. Several studies have used cost for model fitting and / or model evaluation, with various pre-defined cost matrices. Zhou and Liu (2006) found, on 21 UCI data sets and the KDD-99 data set, that using the full data set and adjusting the output for relative costs is more effective than any of the sampling techniques, and the only effective technique on seriously imbalanced data sets. Note that they appear to have used a rather extreme form of over- and under-sampling, where the number of training cases of given class in the sample is proportional to misclassification cost of that class, without regard to the population priors. Kukar and Kononenko (1998), using eleven UCI data sets, and Lawrence *et al* (1998), using a single medical data set, have both found that if costs are known, the most effective strategy of all is to bring cost into the model fitting, either through the objective function or the iterative weight updating. Lawrence *et al* (1998) found under-sampling to be worse than other strategies. Japkowicz and Stephen (2002) created a highly artificial domain – a single input on the interval [0,1], with subdivisions of the interval corresponding to either 100% positive or 100% negative cases, and a greater number of sub-divisions representing a greater complexity. They found that, on small size training sets, over-sampling is better than under-sampling (especially for large class imbalance) which in turn is marginally better than no sampling. They also found that neural networks seem to have larger variance in performance than trees on this domain. A recent study (Mazurowski *et al*, 2008) did not assume any cost structure and used AUC, and another measure called partial AUC, for evaluation. In a study similar in many respects to this one, they considered the effect on model performance of combinations of several factors of interest - sampling strategy, extent of class imbalance and number of cases in the data set - along with the 'confounding factors' type of neural network (backpropogation or swarm optimisation), number of inputs and whether or not inputs are correlated. They used both an artificial domain of two multivariate normal distributions with different means for each class and one medical data set. They found that increasing extent of class imbalance has an increasingly detrimental effect on both average and variance of AUC, under-sampling is worse than either over-sampling or using the full data set, but there is no clear winner between over-sampling and using the full data set.

Numerous studies have used trees as the modelling tool. Again, the results are not entirely consistent, though none of the sampling strategies has been found to be conclusively superior to using the full data set on any of the studies listed. Scarpa and Torelli (2004) used boosted and unboosted trees on the COIL Challenge 2000 data set and one artificial data set, and found, on several evaluation criteria, that "the benefits of sampling are unclear even after applying reasonable corrections". McCarthy, Zabar and Weiss (2005) used C5.0 on twelve data sets from the UCI repository and two from AT&T. They reached no general conclusions, as each sampling method performed best on some of the data sets when evaluated using several cost ratios. Drummond and Holte (2003) applied C4.5 to three UCI data sets and one due to Kubat, Holte and Matwin (1998). They evaluated sampling strategies by using cost curves (an alternative representation of ROC curves) to get an idea of sensitivity of the sampling strategies to differing relative cost assumptions. They concluded that over-sampling is ineffective, under-sampling gives "a reasonable sensitivity to changes in misclassification costs and class distribution", and using the full data set and down-weighting the common class gives the best performance overall. Japkowicz and Stephen (2002) applied C5.0 to the highly artificial domain described in the previous paragraph. They evaluated sampling strategies on both accuracy and total misclassification cost using relative misclassification costs corresponding to the class imbalance. They concluded that only a combination of small sample size, large complexity and great imbalance causes problems, i.e. the entire imbalance issue boils down to essentially a 'small sample' issue – one in which the 'positive regions' consist of only a few training cases. They found that using the full data set and adjusting the output for costs seems to do better than sampling. Chawla (2003) considered C4.5 with and without pruning on four UCI data sets and one data set from University of South Florida. He found that "over-sampling usually leads to small, very specific decision regions".

Very little published research has considered the effect of sampling ratio using logistic regression. Georges (2004) compared models fitted using various sampling ratios on artificially constructed data using mean square bias as an evaluation criterion, and concluded that sampling ratio has relatively little effect on mean square bias.

Of the studies using other modelling methods, Ling and Li (1998) compared different rates of over-sampling the majority class or under-sampling the minority class on three proprietary data sets using boosted Naive Bayes as the modelling method and area under lift curve as the evaluation criterion. They concluded that both over-sampling and under-sampling provide improvement over using the full data set, with the optimal ratio being 1. However, "over-sampling does not provide significant improvement [compared to under-sampling] using the

Naive Bayes algorithm". Japkowicz and Stephen (2002) applied Support Vector Machines to the highly artificial domain described, and found that "for classifiers insensitive to the class imbalance problem, sampling methods usually do not help and may even hurt the classifier performance".

Apart from the studies mentioned above, the largest empirical study has been conducted by Van Hulse, Khoshgoftaar and Napolitano (2007). They considered 35 data sets, seven sampling techniques, eleven modelling algorithms and six evaluation criteria. They generally found both under- and over-sampling to perform better on average than no sampling, although for most of the modelling methods considered this improvement was only statistically significant on evaluation criteria based on an implicit 50% threshold. These results were found even in the case of logistic regression. Moreover, random under-sampling was found to give the best average performance for a greater number of data set, algorithm and evaluation criterion combinations than any other sampling technique. It is not clear why the results are opposite to those of the majority of the other empirical studies done. The authors acknowledge that they have allowed both under- and over-sampling to have a 'free parameter' (degree of under- or over-sampling) , decided after examining the results, which 'no sampling' does not, and that this may be a threat to the validity of the conclusions reached. Further, variability of performance was not considered.

The same research team has also studied the effect of imbalance level on AUC when the number of positive cases available is very small (≤ 40), and concluded that a sampling ratio of 2 or 3 is optimal in such circumstances (Khoshgoftaar *et al*, 2007).

2.5 Gaps in the Literature

There are a number of gaps that can be noted in the literature:

- Very few published studies have systematically varied the sampling ratio of common to rare classes to see whether there is a monotonic relationship of performance (however measured) with sampling ratio. Those that have include Weiss and Provost (2003) and Mazurowski *et al* (2008)[1]. However, these studies were performed under the conditions of a fixed total sample size. In the case of Weiss and Provost this size was equal to the total number of positives in the available training data, with greater sampling ratios formed by removing

[1] Van Hulse, Khoshgoftaar and Napolitano (2007) also considered several sampling ratios, but did not report on the trends observed.

positives from the training data. In the case of Mazurowski *et al* this was achieved via simulation for the artificial data set and via removal of positives for the 'real' data set. Not surprisingly therefore, sampling ratios more balanced than the natural distribution showed better results in both studies. These studies do not, however, shed any light on what the optimum sampling ratio will be when the total number of positives rather than the total sample size is fixed. Georges (2004) and Khoshgoftaar *et al* (2007) *have* varied sampling ratio while keeping total number of positives fixed. However George's study is only a very preliminary study on artificially generated data sets in a set of course notes which does not appear to have been followed up by an independent journal or book publication, while Khoshgoftaar *et al* consider only data sets having a very small number (≤ 40) of positive cases.

- Very few studies have considered a variety of evaluation measures for the same model fit. Empirical studies have tended to use either classification accuracy, measures based on sensitivity and specificity at a 50% cutpoint, or measures of quality of ranking (Kotsiantis, Kanellopoulos and Pintelas, 2006). Very few have considered measures of accuracy of the numerical values of the predicted probabilities.
- Although some studies have used methodologies such as cross-validation to get a more robust measure of average performance of a particular sampling strategy, very few have done any investigation into the variability of the results produced from sampling. This is important, since even if on average we get a similar performance from different sampling strategies, the performance of any given single fitting on unseen data may be very poor if the variability is high.
- Few papers have explored the interaction between model type and sampling ratio on performance. Very few, if any, have explored the effect of this interaction on the variability of performance.

This research attempts to shed some light on the gaps mentioned.

3 Methodology

This research investigates empirically the interaction between the ratio of common class to rare class cases in the training data (sampling ratio) and model type on model performance, and variability of performance, for prediction of a binary target for which one outcome is rare. It investigates whether there are monotonic relationships between the sampling ratio used in the training sample and both the variability of performance and the mean of performance, and whether these relationships depend on whether a ‘flexible’ or ‘inflexible’ model type is used. Relating to this is the question of how model performance is measured, and the research investigates whether the conclusions change depending on what evaluation criterion is used. In particular, it investigates whether evaluation criteria that depend on the ranking of the predicted probabilities only show a different pattern to those that depend on the actual numerical values of the predicted probabilities and are therefore more sensitive to overfitting.

3.1 Hypotheses

The hypotheses tested state broadly that the common practice of under-sampling the common class has a negative effect on the performance of models, particularly for flexible model types when evaluated using criteria sensitive to overfitting. Specifically the hypotheses are:

In a practical context, for which the number of rare class cases available for modelling is limited:

1. The common practice of under-sampling the common class increases the variability of model performance, worsens the mean of model performance, or both, when applied to unseen data, compared to training using all available cases of both classes, and that these effect(s) are more severe:
    a) for flexible modelling methods than for inflexible methods.
    b) for lower sampling ratios in the training sample than for higher ratios.
2. The effect(s) hypothesised in 1. above are greater when model performance is evaluated using criteria that are more sensitive to overfitting than criteria less sensitive to overfitting.

3.2 Modelling Methods

In this study logistic regression was used as the 'inflexible' modelling method and multilayer perceptron neural network (MLP) as the 'flexible' modelling method. Both have parametric or 'global' model forms, i.e. all of the parameters are fit simultaneously on the training data so as to optimise an appropriate objective function. For the MLP, two model sub-types were considered:

a) without early stopping, i.e. the final weights at convergence on the training set were selected.
b) with early stopping. In this implementation, the model was trained until convergence on the training data set, and the weights at the iteration of the training process giving best performance on the validation data set were selected.

Split-sample validation as described in Section 3.3 was used, and for both logistic regression and neural network with early stopping the validation set was used to choose among candidate models.

In the case of logistic regression, forward stepwise selection was used, with a significance level for entry of 0.05. The final model chosen was that at the step which gave minimum deviance on the validation set. Forward selection was used so as to make the chosen model more robust to overfitting than simply including all inputs.

For the neural networks, the following settings were used:

- Single hidden layer.
- Number of hidden units used varied from data set to data set and increased with increasing number of rare class cases. Existing literature on the data set was considered, and some experimentation was performed.
- Inputs standardised to have a mean of zero and a standard deviation of one.
- Objective function - deviance (minimised).
- Linear combination function and hyperbolic tangent activation function for the hidden layer.
- Linear combination function and logistic activation function for the output layer.
- Conjugate gradient descent. Although quasi-Newton may be more efficient for networks with a moderate number of weights (Fletcher, 1987), conjugate gradient descent tends to take smaller steps around the parameter space, which is preferable for early stopping (Potts, 2000). The conclusions reached did not change when quasi-Newton was tried.

- Initialisation: five preliminary runs, each from a different random starting point, were each run for three iterations of conjugate gradient descent. It was found that longer preliminary runs could overfit the validation data in some cases, i.e. the validation error could start to rise before the end of the preliminary run. Preliminary runs are useful to try and avoid finding local minima. The weights corresponding to the final iteration of the preliminary run giving best performance on the training data set were used to initialise actual training. The default SAS PROC NEURAL settings were used to initialise each preliminary run. These are:
    - Input-to-hidden weights generated from the standard normal distribution.
    - Output bias set to the target mean, transformed by the logit link function.
    - Hidden-to-output weights set to zero.
- Convergence criteria: SAS PROC NEURAL defaults were used.
- In the case of early stopping, the chosen weights could be those for either the start or end point of one of the five preliminary runs or any iteration of the actual training run.

Maximum likelihood estimation was used for both model types, as this is by far the most common method used in practice. Bernoulli likelihood was used for the binary target.

3.3 Sampling Methods and Model Evaluation

Split-sample validation was used. Cross-validation was avoided, as each iteration of the cross-validation procedure results in only a single measure of model performance, so many more runs would have been required. The available data set was partitioned via random sampling stratified on the target into a training subset T, a validation subset V and a test subset E. The stratification ensured the same proportion of positives and negatives in all three subsets. In the case of the real-world data sets studied, the partition was 40% training, 30% validation and 30% test. In the case of the artificial data sets, the partition was 10% training, 9% validation and 81% test. This is because for the artificial data the number of observations, and hence the test set, could be made arbitrarily large by simulating more observations.

After partitioning, the actual training set T^* was created by including all of the N_1 positive observations in T and (sampling ratio)* N_1 negative observations sampled randomly from T. The same procedure was applied to create the validation set V^* from V. E was used as the actual test set, i.e. the prior probabilities were maintained in the test set.

For each model fit, the test set was scored, and the predicted probabilities on the test set were corrected for under-sampling of the common class to population predicted probabilities using the formula (Georges, 2004):

$$\hat{p}_1 = \frac{\tilde{p}_1(\pi_1/\rho_1)}{\tilde{p}_0(\pi_0/\rho_0) + \tilde{p}_1(\pi_1/\rho_1)}$$

where

$\hat{p}_1$	is the population predicted probability of class 1
$\tilde{p}_0$ and $\tilde{p}_1$	are the unadjusted predicted values based on the training set sampling ratio
π_0 and π_1	are the population priors for class 0 and 1 respectively
ρ_0 and ρ_1	are the overall proportions of class 0 and 1 in the training data.

This is equivalent to correcting the intercept term (hidden-to-output bias in the case of the neural network) by subtracting $\ln(\frac{\pi_0 \rho_1}{\pi_1 \rho_0})$, as discussed in Section 2.2.3.

Four evaluation criteria were calculated on the test set. All are invariant under differing cost assumptions. These four were: Area under the ROC curve (AUC), proportion of positives in top 10% of predicted probabilities, mean squared deviance residual, and mean square error (MSE). MSE is the mean of squared error loss as defined in Section 2.3.2. The first two criteria are based on the rankings of the predicted probabilities while the last two are based on the numerical values of the predicted probabilities.

After the results for the artificial data sets were analysed, it was decided to use only two evaluation criteria on the real-world data sets: AUC and MSE.

3.4 Significance Testing

The sampling ratio was considered as a classification variable with number of levels dependent on the degree of class imbalance in the data set considered. For the artificial data sets it had five levels: 1, 2, 4, 8 and 16, representing ratios of 1:1, 2:1, 4:1, 8:1 and 16:1 respectively. Since the distribution of some of the evaluation criteria at some of the sampling ratios was highly skewed, and the usual F-test for equality of variances is very sensitive to departures from the assumption of normal distribution for both groups being compared, Levene's test (Levene, 1960), which performs an analysis of variance (ANOVA) F-test on the absolute value of deviations from each group mean, was used to test for equality of variance among groups. Welch's variance-weighted ANOVA (Welch, 1951), which is robust to the assumption of equal within-group variances, was used to test for equality of means. It should be noted that the observations (run results) are not completely independent, since there is some degree of overlap between the training cases in each run, but this is not believed to affect the conclusions reached to a marked degree. If the overall comparison of all five groups (sampling ratios) showed significance at the 5% level, then the tests were performed pairwise on ratios 1 vs. 2, 2 vs. 4, 4 vs. 8 and 8 vs. 16 to determine until what ratio a significant difference was found. Only differences significant at the 5% level are reported.

3.5 Data Sets

Six data sets were used, three artificial and three real-world. These data sets are described in the relevant results chapters.

3.6 Experimental Design

The experimental design is shown algorithmically below. The design takes into account both random aspects associated with the common data mining practice described in Chapter 1:

a) Random partitioning of data into training, validation and test sets.
b) Random selection of negative training and validation cases from their respective partitions for a given sampling ratio.

The number of runs (30) was chosen so as to detect a reduction in variance of 50% at the 5% significance level using a 1-sided F-test (minimum runs required is 25). Due to non-normality of the data, however, the tests mentioned in Section 3.4 were used instead.

Experimental Design

```
For data set = $D_1$ to $D_6$:

-   Pre-select relevant variables using existing literature on the data set and class
    distribution trees.

-   For run = 1 to 30:

        o   Partition the data into training, validation and test subsets $T$, $V$ and $E$ using
            random sampling stratified on the target to ensure the same target class
            proportions in $T$, $V$ and $E$.

        o   For $N_0$:$N_1$ ratio = 1,2,4,8,16,…,'full data set'

                ▪   Create the actual training set $T*$ by including all of the rare class
                    cases in $T$ and a simple random sample of size (($N_0$:$N_1$ ratio) * $N_{1T}$)
                    from the $N_{0T}$ common class cases in $T$, where $N_{1T}$ is the number
                    of rare class cases in $T$.

                ▪   Create the actual validation set $V*$ by including all of the rare class
                    cases in $V$ and a simple random sample of size (($N_0$:$N_1$ ratio) * $N_{1V}$)
                    from the $N_{0V}$ common class cases in $V$.

                ▪   Impute missing values using the training set mean for interval inputs
                    and most frequent class for categorical inputs. Missing value
                    imputation was not the focus of this study so simple methods were
                    used.

                ▪   For model = Logistic Regression, Neural Net, Neural Net with early
                    stopping:

                        ♦   Fit the model

                        ♦   Score the test set $E$ and adjust the predicted probabilities
                            using the formula in Section 3.3.

                        ♦   Evaluate the fit on test set $E$ using the evaluation criteria
                            described in Section 3.3.

                ▪   End for model

        o   End For $N_0$:$N_1$ ratio

-   End For run

-   Calculate the mean and the standard deviation of the assessment measure(s) per $N_0$:$N_1$
    ratio.

End for data set
```

4 Results – Artificial Data

Before proceeding to examine the results on some real-world data sets, the results from some simulated data are discussed. Simulated data have the advantages that:

a) the actual probabilities for class 1 and class 0 can be determined analytically, and hence predicted probabilities can be compared with actual probabilities.
b) the test set can be made large enough to be truly representative of the population.
c) insights can be gained that can be applied when we come to look at real-world data.

To avoid confusion with an iteration of conjugate gradient descent, a single iteration of the entire training procedure will be referred to as a 'run'. A training set having a 1:1 ratio of common to rare class cases will be referred to as having a 'lower' sampling ratio than one having, for example, a 16:1 ratio of common to rare class cases. 'Regression' always refers to logistic regression.

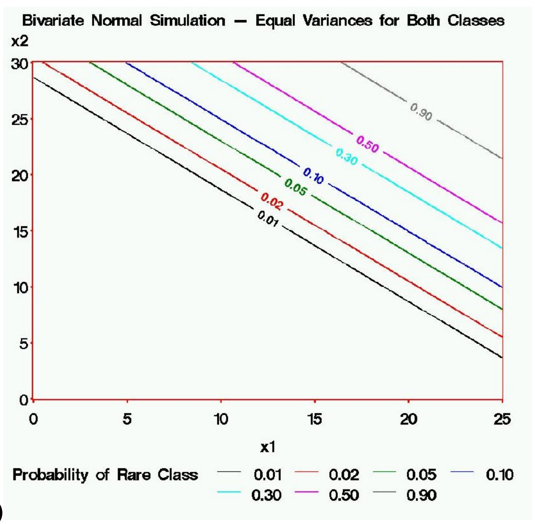
4.1 Simulation Methodology

Data was simulated for three bivariate normal scenarios for the inputs. In each scenario, the correlation between the two inputs X_1 and X_2 was 0.3, the means were 13 and 18 respectively for the common class and 15 and 20 for the rare class, and the variance for the common class was 4 for both inputs. The variances of the inputs for the rare class were altered in the different scenarios: 2 and 2 in the 'smaller variances for rare class' scenario, 4 and 4 in the 'equal variances for both classes' scenario and 8 and 8 in the 'larger variances for rare class' scenario. For any given $\mathbf{x} = (x_1, x_2)$ we can use Bayes's rule to calculate the actual probability of class 1 $p_1(\mathbf{x})$ as:

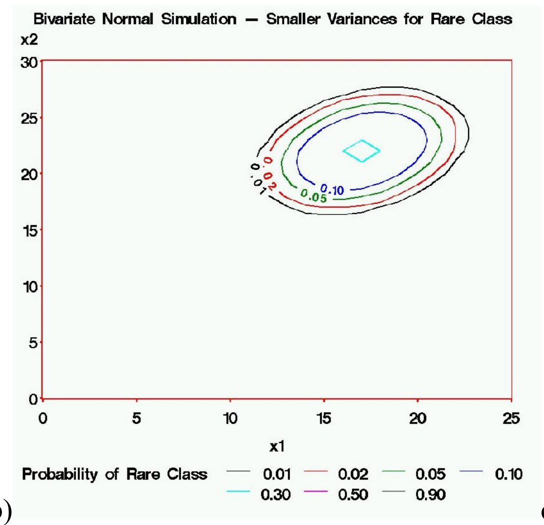
$$p_1(\mathbf{x}) = \pi_1 f_1(\mathbf{x}) / \{\pi_0 f_0(\mathbf{x}) + \pi_1 f_1(\mathbf{x})\},$$

where π_0 and π_1 are the population priors and $f_0(\mathbf{x})$ and $f_1(\mathbf{x})$ are the bivariate normal densities for class 0 and 1 respectively.

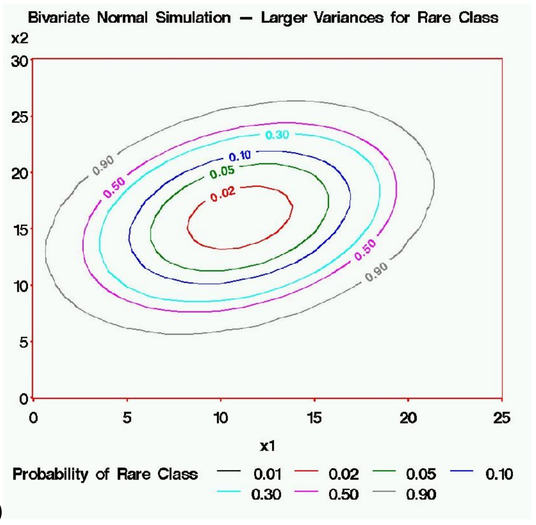
Figure 4.1 shows contour plots of $p_1(\mathbf{x})$ in each of the three scenarios.



a)



b)



c)

Figure 4.1: Contour plots of actual probability of the rare class for three bivariate normal simulations, each with a prior of 0.05. Class 1 has a mean of 15 for X_1 and 20 for X_2. Class 0 has a mean of 13 for X_1 and 18 for X_2. Correlation between X_1 and X_2 is 0.3. Where the variances of the inputs are equal for both classes (a), the contour lines are linear and parallel, otherwise they are non-linear. In b), the probability of class 1 reaches a maximum of 0.329 when X_1=17 and X_2=22. In c), the probability of class 1 reaches a minimum of 0.012 when X_1=11 and X_2=16. In a) the probabilities are bounded by 0 and 1 only.

For each run, 1000 observations from the distribution for positives and 19000 from the distribution for negatives were simulated, i.e. a prior of 0.05 was used. Five sampling ratios were considered in each scenario: 1, 2, 4, 8 and 16. Thirty runs were performed at each ratio in each scenario. For each run, the data set was partitioned in proportions 10:9:81 into the training, validation and test subsets using random sampling stratified on the target, i.e. each training set contained 100 positive cases and each validation set contained 90 positive cases. Each test set consisted of 810 positive and 15390 negative cases, i.e. the prior class distribution was maintained in the test set.

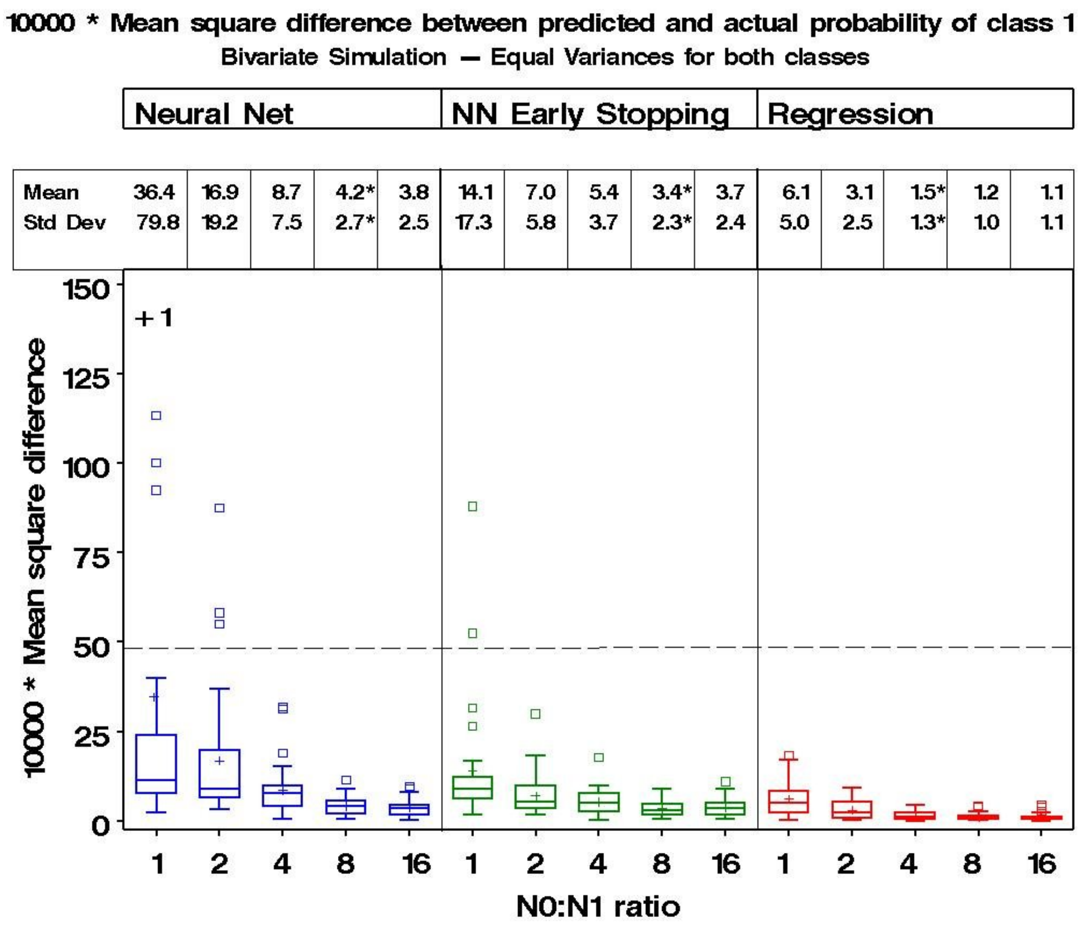
Two hidden units were used for the neural networks. The neural networks had nine parameters and the regression three parameters to be estimated. A rough rule of thumb for logistic regression (Peduzzi *et al*., 1996) is that there should be at least ten observations of each class for each parameter to be estimated. Although there is much disagreement as to such rules of thumb for neural networks (Sarle, 1997), in this simulation we have approximately 11 positive observations per parameter to be estimated for the neural networks and 33 for the regressions. Despite the fact that the data are probably more dense in the input space than we are likely to encounter in practice, relevant insights are provided.

4.2 Comparison of Predicted and Actual Probabilities of Rare Class

Since the data have been simulated, for each observation $\mathbf{x} = (x_1, x_2)$ in the test data set for a given partitioning run, we can calculate the actual probabilities for class 0 and 1 as described in Section 4.1. Hence we can calculate the squared difference between the actual and predicted probability of class 1 for each observation, and averaging these over the test set provides a measure of how well the model has approximated the true probabilities for that run. Since it is only on simulated data that this can be done, the results will be discussed in some detail.

Figures 4.2 to 4.4 show box-and-whisker plots for mean squared difference between actual and predicted probability of class 1 (which will be referred to as MSD) for each scenario. All squared differences have been multiplied by 10000 to make the results easier to read. Outliers are defined here as runs producing MSD values more than 1.5 times the inter-quartile range above the 75th percentile or below the 25th percentile, and are indicated by squares on the box plots. The baseline – MSD achieved by assigning the prior probability to all observations – is shown by a dashed horizontal line. Its calculation is described in Section 4.5.

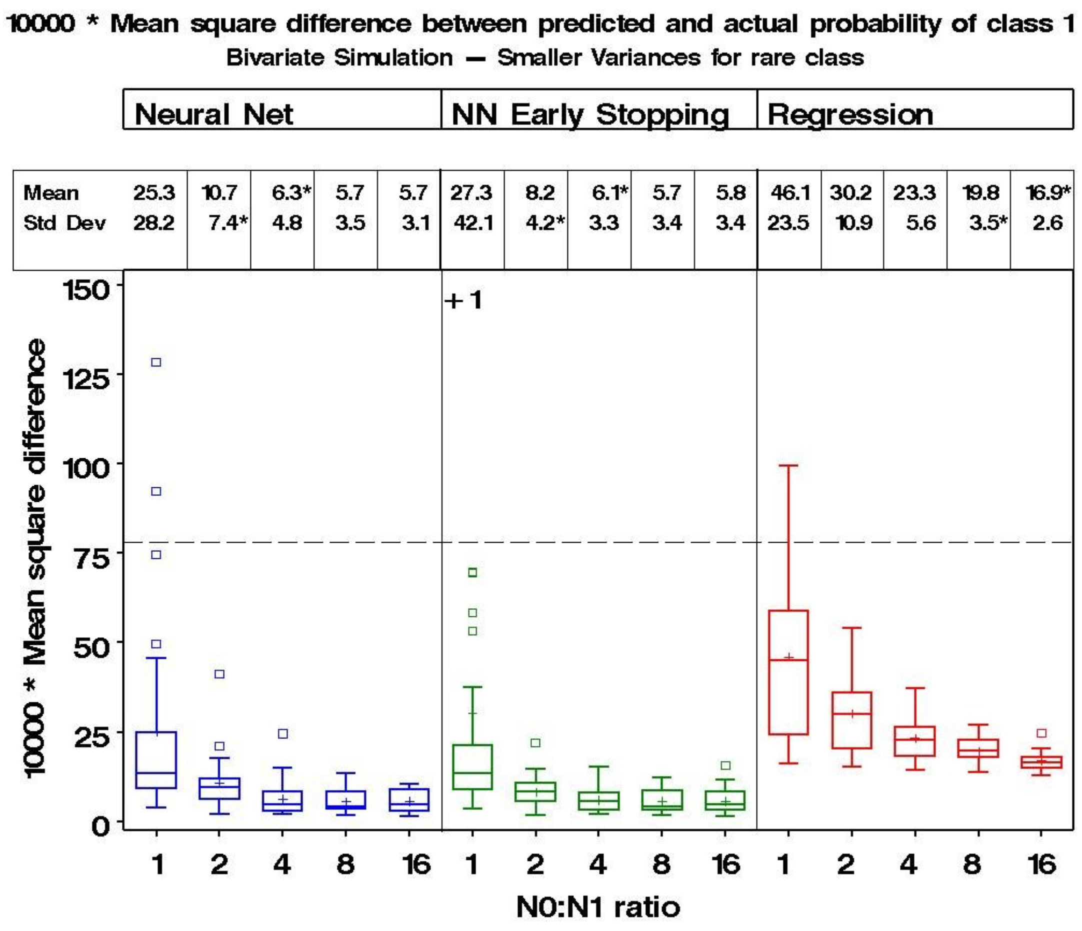
Consider first the scenario in which the inputs have equal variance for both classes, shown in Figure 4.2. In this case, the probability contours are linear and parallel, and in fact linear discriminant analysis provides the analytically optimal solution (Hastie, Tibshirani and Friedman, 2001). Thus we would expect regression to perform at least as well as the neural networks, and indeed both the variance and the mean of MSD are lower for regression than for the neural networks at every sampling ratio. Nevertheless, for regression there is still



	Neural Net					NN Early Stopping					Regression				
	1	2	4	8	16	1	2	4	8	16	1	2	4	8	16
Mean	36.4	16.9	8.7	4.2*	3.8	14.1	7.0	5.4	3.4*	3.7	6.1	3.1	1.5*	1.2	1.1
Std Dev	79.8	19.2	7.5	2.7*	2.5	17.3	5.8	3.7	2.3*	2.4	5.0	2.5	1.3*	1.0	1.1

Figure 4.2: Mean square difference between predicted and actual probability of class 1 for bivariate normal simulation with equal input variances for both classes. Each box plot is based on 30 observations (runs). An asterisk next to a "Mean" or "Std Dev" figure indicates a statistically significant improvement (at the 5% level) for the corresponding ratio compared to all lower ratios, but no further improvement compared to higher ratios. For example, for regression the mean of 1.5 at ratio 4 is significantly lower than the means of 6.1 and 3.1 at ratios 1 and 2, but not significantly higher than the means of 1.2 and 1.1 at ratios 8 and 16. The dashed horizontal line indicates the value that would be obtained by allocating the prior probability to every case. "+x" indicates x outliers beyond the axis range shown.

a significant reduction in both variance and mean from a ratio of 1 up until a ratio of 4, although the coefficient of variation at ratio 1is relatively small (1.46%). For neural network both with and without early stopping, both the mean and the variance decrease to a ratio of 8, and there are a number of outliers – runs for which the model does a very poor job of approximating the true probabilities on the test data – particularly at ratios of 1 and 2.



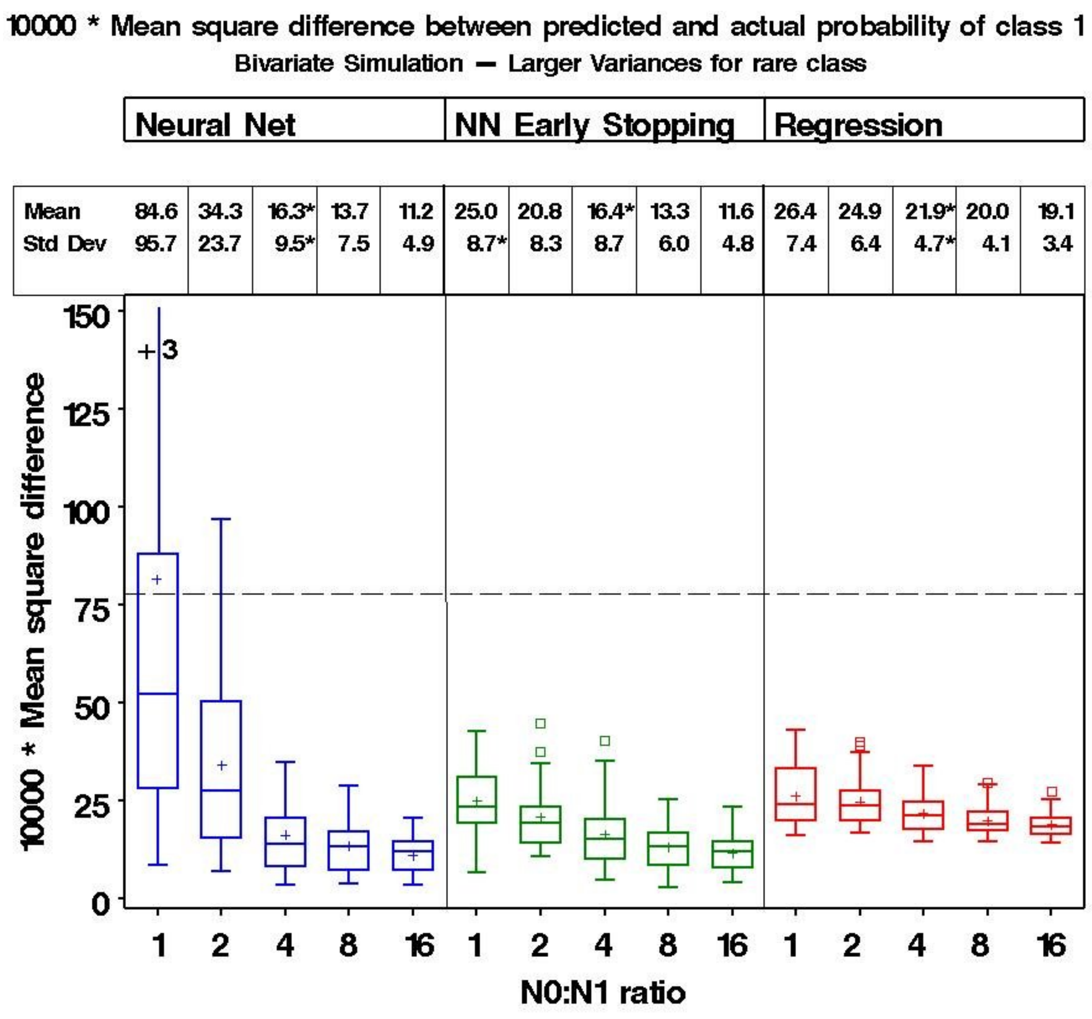
	Neural Net					NN Early Stopping					Regression				
Mean	25.3	10.7	6.3*	5.7	5.7	27.3	8.2	6.1*	5.7	5.8	46.1	30.2	23.3	19.8	16.9*
Std Dev	28.2	7.4*	4.8	3.5	3.1	42.1	4.2*	3.3	3.4	3.4	23.5	10.9	5.6	3.5*	2.6

Figure 4.3: Mean square difference between predicted and actual probably of class 1 for bivariate normal simulation with smaller input variances for the rare class.

The outliers for neural network without early stopping are caused by overfitting the training data – predicted probabilities very close to 1 or 0, and actual probabilities far smaller or larger respectively. This occurs on training samples for which the region of larger values of both X_1 and X_2 contains only positives and the region of smaller values of both X_1 and X_2 contains only negatives. For neural network with early stopping the outliers are caused by overfitting the validation data, and usually occur when training is stopped after a very small number of iterations – i.e. the final weights from one of the preliminary runs, or a very early iteration of the actual training run, perform best on the validation data, but not on the test data. Since the validation data has 90 positive cases and 90% of the number of negative cases in the training data, it may not be representative of the underlying distribution. In summary, in this scenario it is true that the variance of the accuracy of estimation of the class probabilities decreases with increasing sampling ratio, and that this effect is greater for the flexible modelling method than for the inflexible modelling method.

Considering next the scenario in which the inputs have smaller variance for the rare class than for the common class (Figure 4.3), we see a very different picture. There are still some outliers – on training samples for which the region of smaller values of both X_1 and X_2 contains only negatives – for neural network without early stopping, and also some with early stopping at a ratio of 1, but from a ratio of 2 upwards the neural networks outperform the regression in both mean and variance of MSD. Moreover, for regression there is a decrease in mean all the way to a ratio of 16 and in variance to a ratio of 8, while for the neural networks there is only a decrease to a ratio of 4 in mean and to a ratio of 2 in variance. Since the means for both inputs X_1 and X_2 are larger for the rare class than the common class, the regression coefficients are always positive. Hence the regression cannot 'find' the fact that the probability of the rare class does not increase monotonically with increasing X_1 and X_2, but starts decreasing for values of (X_1, X_2) greater than (17,22). The larger the sampling ratio, the greater, on average, the number of negative cases in this region and therefore the more stable the regression coefficients (the standard deviation of the coefficient for X_1 decreases from 0.11 to 0.05, and that for X_2 from 0.14 to 0.05 from a ratio of 1 to a ratio of 16). Thus while it is true that in this scenario the variance of the accuracy of estimation of the class probabilities decreases with increasing sampling ratio, it is not true that this decrease is greater for the more flexible modelling method than the less flexible method; in fact the reverse is true.

Considering lastly the scenario in which the inputs have larger variance for the rare class than for the common class (Figure 4.4), we see that neural network without early stopping gives a very large variance of MSD at a ratio of 1 with many outliers. These occur on training samples having a pure positive region of larger values of both X_1 and X_2. Both the variance and the mean decrease sharply to a ratio of 4. With early stopping, Levene's test shows no significant decrease in variance with increasing ratio, although the single comparison of variance at ratio 16 versus ratio 1 is significant. There is a decrease in mean to a ratio of 4. For regression, there is a decrease in both variance and mean until a ratio of 4. As in the 'smaller variances for rare class' scenario, because the probability of the rare class is not monotonically related to the inputs – for example from the point X_1=11 and X_2=16, the probability of a positive outcome increases with either increasing or decreasing X_1 or X_2 – the regression cannot achieve the mean accuracy of, and has similar variance to, the neural networks at higher sampling ratios. We can conclude that in this scenario increasing the sampling ratio does provide some improvement in either the variance or the mean of the accuracy of estimation of the class probabilities up to a ratio of 4. Further, in this scenario the reduction in variance is greater for the more flexible modelling method than for the less flexible method only if no regularisation (early stopping) is used for the more flexible method.



	Neural Net					NN Early Stopping					Regression				
N0:N1 ratio	1	2	4	8	16	1	2	4	8	16	1	2	4	8	16
Mean	84.6	34.3	16.3*	13.7	11.2	25.0	20.8	16.4*	13.3	11.6	26.4	24.9	21.9*	20.0	19.1
Std Dev	95.7	23.7	9.5*	7.5	4.9	8.7*	8.3	8.7	6.0	4.8	7.4	6.4	4.7*	4.1	3.4

Figure 4.4: Mean square difference between predicted and actual probability of class 1 for bivariate normal simulation with larger input variances for the rare class.

Note that in the 'larger variances for rare class' scenario, none of the techniques can approximate the actual class probabilities as accurately as in the other two scenarios. This is because the high probability region for class 1 for low X_1 and / or low X_2 is a low density region, and it is very likely that there are no positive observations in this region in the training sample. Thus even the neural networks cannot accurately 'find' this region.

4.3 Real-world Evaluation Criteria Applied to the Artificial Data

We now evaluate model performance on the artificial data using criteria that can be applied to real-world data, for which we do not know the true probabilities of the rare class given the inputs.

First it should be noted that for none of the scenarios considered, and for none of the evaluation criteria considered, was there an increase in variance or a worsening of mean with increasing sampling ratio. In other words, for none of the criteria considered was there a worsening of performance with increasing sampling ratio. So we need only consider whether there was a *statistically significant* reduction in variance, improvement in mean or both with increasing sampling ratio, and if so, up until what ratio this was significant. Note that the word 'improvement' will be used to mean a reduction in variance on any criterion, an increase in mean on Area under the ROC curve (AUC) or proportion of positives in top 10% of predicted probabilities, and a decrease in mean on mean squared deviance residual or mean square error (MSE). Improvement will also generally be referred to as improvement relative to the 1:1 sampling ratio. Note also that the mean squared deviance residual criterion will be referred to simply as 'deviance'.

Table 4.1 summarises this information for each of the three scenarios.

4.3.1 Ranks-based Criteria

It can be seen from Table 4.1 that the two ranks-based criteria, AUC and proportion of positives in top 10% of predicted probabilities (which will be referred to as Prop1Top10%), give similar results to one another in all three scenarios, with Prop1Top10% being slightly more sensitive to overfitting. For regression, both AUC and Prop1Top10% show no significant improvement in either variance or mean with increasing sampling ratio in any of the scenarios. For neural network with early stopping there is no improvement in variance or mean with increasing ratio in the 'equal variances for both classes' and 'larger variances for rare class' scenarios. In the 'smaller variances for rare class' scenario there is an improvement in both variance and mean to a ratio of 2 on the Prop1Top10% criterion and an improvement to a ratio of 2 in mean only on the AUC criterion. For neural network without early stopping, an improvement to a ratio of 2 in both variance and mean is found in most cases, except that an improvement in mean to greater ratios can be noted in the 'equal variances for both classes' scenario. In summary, on the ranks-based criteria, except in the case of equal variances and neural network without early stopping, a sampling ratio of 2 cannot be improved upon.

4.3.2 Criteria Based on Numerical Values of Predicted Probabilities

The results for the two criteria based on the values of the predicted probabilities, MSE and deviance, are also similar to one another. For regression, a reduction in both mean and

Input Variances of Rare Class Relative to Common Class	**Criterion**	**Method**	**Decrease in Variance from ratio 1 until Ratio**	**Improvement in Mean from ratio 1 until Ratio**
EQUAL	AUC	NN	2	4
		NN_ES	-	-
		Reg	-	-
	Prop1top10%	NN	2	8
		NN_ES	-	-
		Reg	-	-
	Deviance	NN	8	4
		NN_ES	2	2
		Reg	2	2
	MSE	NN	8	4
		NN_ES	2	2
		Reg	2	2
SMALLER	AUC	NN	-	2
		NN_ES	-	2
		Reg	-	-
	Prop1top10%	NN	2	2
		NN_ES	2	2
		Reg	-	-
	Deviance	NN	2	4
		NN_ES	2	2
		Reg	8	4
	MSE	NN	2	2
		NN_ES	2	2
		Reg	4	16
LARGER	AUC	NN	-	-
		NN_ES	-	-
		Reg	-	-
	Prop1top10%	NN	2	-
		NN_ES	-	-
		Reg	-	-
	Deviance	NN	4	2
		NN_ES	-	4
		Reg	-	4
	MSE	NN	8	2
		NN_ES	8	16
		Reg	4	4

Table 4.1: Summary of results on the four evaluation criteria. 'NN' indicates neural network without early stopping, NN_ES with early stopping, and 'Reg' regression.

variance to a ratio of 2 is found in the ‘equal variances for both classes’ scenario. In the ‘larger variances for rare class’ scenario, an improvement in mean to ratio 4 is found on both criteria, but an improvement in variance to ratio 4 is found on the MSE criterion only. In the ‘smaller variances for rare class’ scenario, the Levene and Welch ANOVA F-values for comparison of both variances and means are very large, indicating a significant decrease in variance to a ratio of 8 on the deviance criterion and to a ratio of 4 on the MSE criterion. A reduction in mean to a ratio of 4 is found on the deviance criterion and to a ratio of 16 on the MSE criterion. For neural network with early stopping, in both the ‘equal variances for both classes’ and ‘smaller variances for rare class’ scenarios, improvement in both mean and variance up to a ratio of 2 is found. In the ‘larger variances for rare class’ scenario, slightly differing results are obtained on the MSE and deviance criteria. The deviance criterion shows a decrease in mean to a ratio of 4 and no significant reduction in variance with increasing ratio, while the MSE criterion shows decrease in mean to a ratio of 16 and decrease in variance to a ratio of 8, although this is only marginally significant. For neural network without early stopping, with some slight differences between the two criteria, variance decreases to a ratio of about 8 in the ‘equal variances for both classes’ and ‘larger variances for rare class’ scenarios, and to a ratio of 2 in the ‘smaller variances for rare class’ scenario, while mean decreases to a ratio of 4 in the ‘equal variances for both classes’ scenario and to a ratio of about 2 in the other scenarios.

To summarise, on the criteria based on the numerical values of the predicted probabilities, for neural network without early stopping, increasing the sampling ratio is beneficial as regards variance up to a ratio of 8 in the ‘equal variances for both classes’ and ‘larger variances for rare class’ scenarios and to a ratio of 2 in the ‘smaller variances for rare class’ scenario, and beneficial as regards mean to a ratio of 2 in all scenarios. For neural network with early stopping, increasing the sampling ratio is beneficial as regards variance up to a ratio of about 2 in the ‘equal variances for both classes’ and ‘smaller variances for rare class’ scenarios and only marginally beneficial in the ‘larger variances for rare class’ scenario, and beneficial as regards mean to a ratio of 2 in the ‘equal variances for both classes’ and ‘smaller variances for rare class’ scenarios and to a ratio of 16 in the ‘larger variances for rare class’ scenario. For regression, increasing the sampling ratio is beneficial as regards variance to a ratio of 2 in the ‘equal variances for both classes’ scenario, marginally beneficial to a ratio of 4 in the ‘larger variances for rare class’ scenario and beneficial to a ratio of about 8 in the ‘smaller variances for rare class’ scenario, and similar results apply to the mean.

4.4 Correlations between the Evaluation Criteria and Mean Square Difference between Actual and Predicted Probabilities

It is useful to examine which of the evaluation criteria is most highly correlated with the 'true' accuracy of approximation to the actual probabilities, as given by MSD. Because the Pearson linear correlation is highly influenced by outliers, both the Pearson and Spearman rank correlations are shown in Table 4.2.

Pearson Correlation Coefficients, N = 450					
Variances of Inputs for Rare Class relative to Common Class		**AUC**	**Proportion of 1s in top 10% of predicted probabilities**	**Deviance**	**Mean Square Error**
EQUAL	**Mean of (Actual-Pred P(1))^2**	-0.16376	-0.20999	0.92967	0.99200
SMALLER		-0.49928	-0.44131	0.56493	0.99161
LARGER		-0.14463	-0.10860	0.84202	0.99392
Spearman Correlation Coefficients, N = 450					
EQUAL	**Mean of (Actual-Pred P(1))^2**	-0.28079	-0.22586	0.67874	0.74470
SMALLER		-0.73089	-0.71762	0.92725	0.96226
LARGER		-0.38157	-0.31479	0.81457	0.93260

Table 4.2: Pearson and Spearman correlation coefficients between each of the evaluation criteria and the mean of (Actual – Predicted probability of rare class)^2.

From Table 4.2 it is clear that mean square error is the criterion most the highly correlated with MSD, followed by deviance, while the correlation between the ranking quality criteria (AUC and Prop1Top10%) and MSD is relatively low.

4.5 Correlations among the Evaluation Criteria

Table 4.3 shows the Pearson correlations among the evaluation criteria. It is evident that MSE has a very strong correlation with deviance, and that AUC has a strong correlation with Prop1Top10%. In view of the fact that MSE is more highly correlated with MSD than deviance, and AUC is more highly correlated with MSD than Prop1Top10% and also a more widely recognised measure, only mean square error and AUC will be used for evaluation of the real data sets in Chapter 5.

Pearson Correlation Coefficients, N = 450					
Variances of Inputs for Rare Class relative to Common Class		**AUC**	**Proportion of 1s in top 10% of predicted probabilities**	**Deviance**	**Mean Square Error**
EQUAL	**AUC**	1.00000	0.69933	-0.07778	-0.19825
	Prop1Top10%	0.69933	1.00000	-0.12035	-0.25538
	Deviance	0.07778	-0.12035	1.00000	0.92045
	MSE	0.19825	-0.25538	0.92045	1.00000
SMALLER	**AUC**	1.00000	0.87546	-0.55505	-0.53582
	Prop1Top10%	0.87546	1.00000	-0.48998	-0.47746
	Deviance	0.55505	-0.48998	1.00000	0.56230
	MSE	0.53582	-0.47746	0.56230	1.00000
LARGER	**AUC**	1.00000	0.63494	-0.14963	-0.16668
	Prop1Top10%	0.63494	1.00000	-0.01274	-0.13733
	Deviance	0.14963	-0.01274	1.00000	0.83873
	MSE	0.16668	-0.13733	0.83873	1.00000

Table 4.3: Pearson correlation coefficients among the evaluation criteria.

In Section 4.4 it was shown that MSE has a high correlation with MSD. For interpretation of the practical significance of the results on the MSE criterion obtained on the real data sets in Chapter 5, it is also important to compare the percentage improvements achieved by the best model over the baseline – the performance achieved by assigning the prior probability to every observation – for MSE and MSD. The baseline MSE for a 5% prior is $0.05*0.95^2+0.95*0.05^2=0.0475$. In order to establish a baseline MSD, the squared difference between the actual probability of class 1 and the prior was calculated for each observation in each test set, and the average over each test set was calculated. This average was then averaged over all test sets.

In the ‘equal variances for both classes’ scenario, the best model achieved a 9% improvement (from 0.0475 to 0.0432) in average MSE over the baseline compared to a 98% (from

0.004857 to 0.000109) improvement in MSD. In the ‘smaller variances for rare class’ scenario, a 9% improvement in MSE over the baseline corresponded to an 89% improvement in MSD. In the ‘larger variances rare class’ scenario, a 13% improvement in MSE over the baseline corresponded to an 86% improvement in MSD. It can be seen that relatively small improvements in MSE relative to the baseline translate into far greater relative improvements in the accuracy of estimation of the true probabilities.

This ‘magnification’ effect actually increases with decreasing prior. For example, when the ‘equal variances for both classes’ scenario was re-run with the same number of positives but a greater number of negatives, so as to make the prior 2%, the best model achieved a 4% improvement in MSE but a 97% improvement in MSD.

4.6 Summary

While it has been shown that, in general, an increase in sampling ratio results in a reduction in the variance of performance, an improvement in the mean of performance, or both, the extent of this improvement and its relationship to model flexibility is dependent on the complexity of the underlying probability structure. In the case of a localised region of high probability, a less flexible modelling method can show greater benefit from an increased sampling ratio than a more flexible method. If, on the other hand, the region of high probability is less localised, a more flexible method shows greater benefit from an increased sampling ratio than a less flexible method.

The flexible modelling method has a tendency to produce some very large outliers – i.e. greatly overfit the data for low sampling ratios – particularly a ratio of 1. Hence it is possible to obtain a very poor result for one specific fit of a flexible model at a low sampling ratio. This tendency is negated considerably by the use of early stopping.

The evaluation criteria based on the numerical values of the fitted probabilities show far more sensitivity to overfitting than the measures based on the ranks of the fitted probabilities only.

Of the four evaluation criteria examined, it is sufficient to concentrate on only two: mean square error and area under the ROC curve.

5 Results – Real Data

Three data sets were considered: a proprietary bank home loan data set, the KDD Cup 1998 data set and the COIL 2000 Challenge data set. Since differing results were obtained on each, they are discussed separately. All had a binary target with rare target class and a large number of observations, hence are suitable for the methodology described in Chapter 3.

Following the conclusions reached when examining the artificial data sets in Chapter 4, only two evaluation criteria were used: mean square error (MSE) and Area under the ROC curve (AUC), both calculated on the test set. As was seen in Chapter 4, MSE is the more sensitive to overfitting, as it is based on the numerical values of the predicted probabilities, while AUC is based merely on their ranking.

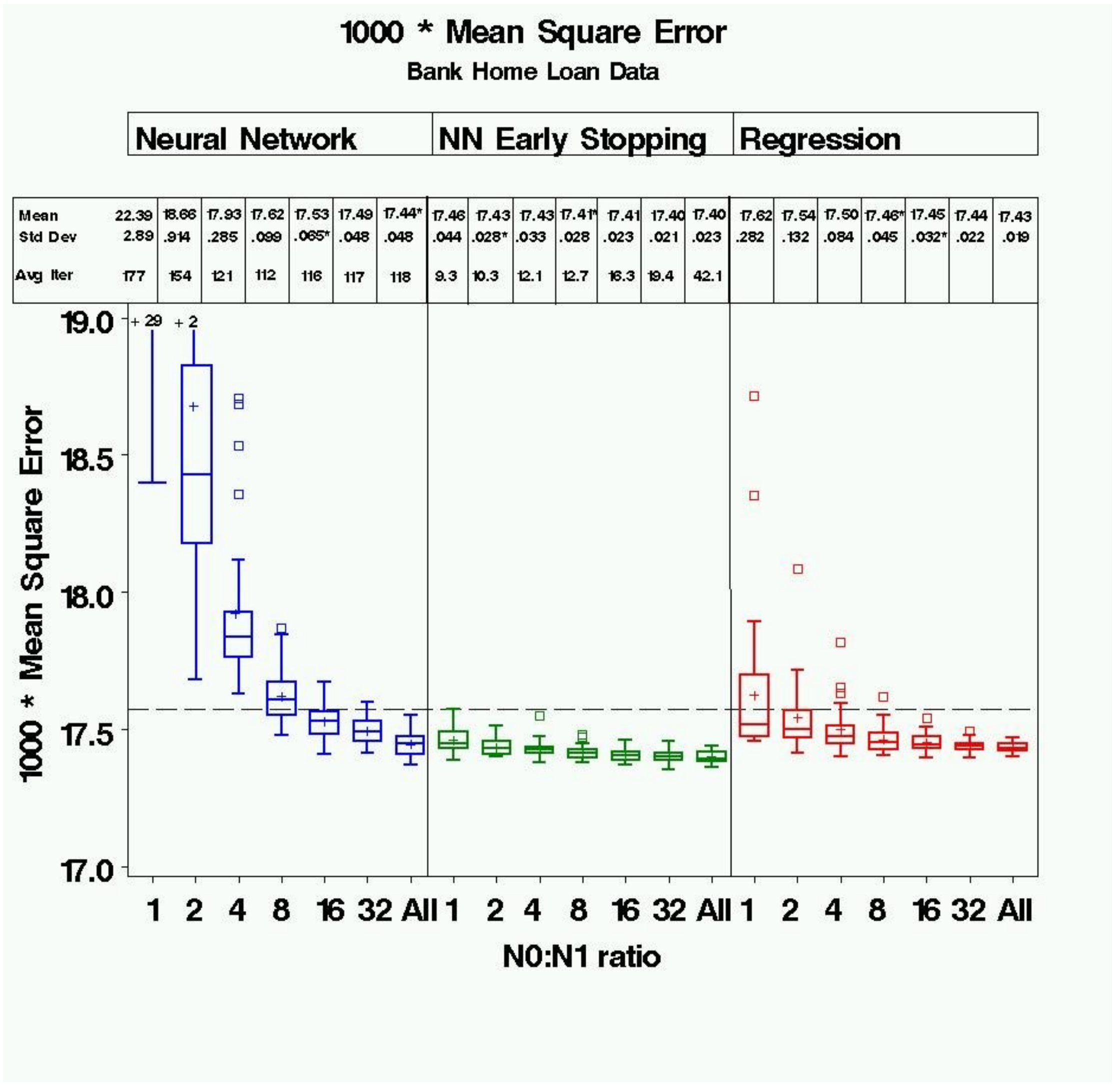
Only results statistically significant at the 5% level, using the tests described in the Chapter 3, are reported.

5.1 Bank Home Loan Data Set

This is a proprietary data set from a large bank. The target was sale of a new home loan in a six month period to qualifying existing bank customers. For this data set, 3142 / 175584 customers had positive response, i.e. the prior was 1.79%. The sampling ratios considered were 1, 2, 4, 8, 16, 32 and 'full data set', where 'full data set' corresponds to a ratio of 54.9.

Thirty-three interval inputs and five categorical inputs were pre-selected for consideration. These are listed in Appendix A. Five hidden units were used for the neural networks.

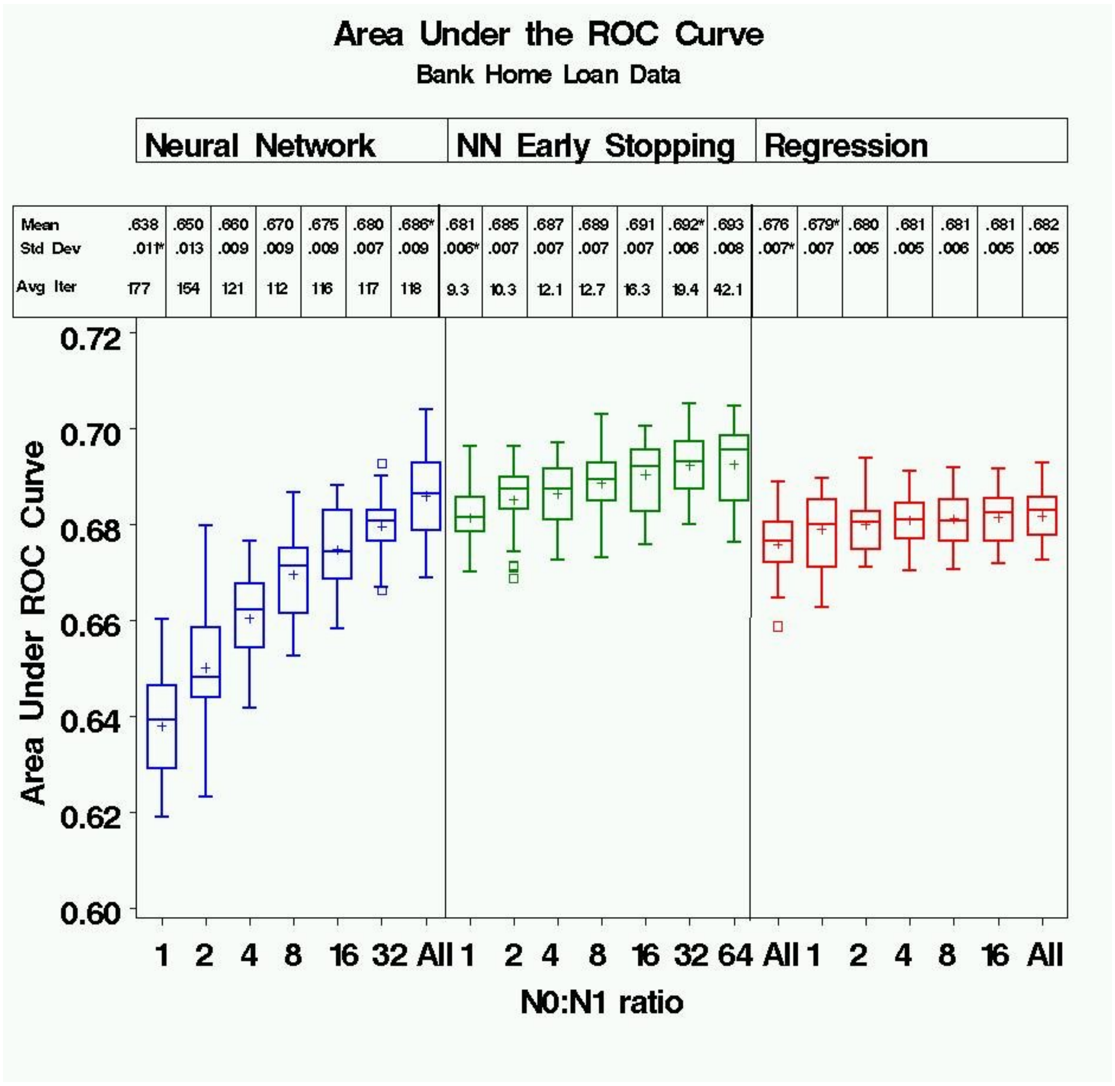
Figure 5.1 shows the box-and-whisker plot for MSE and Figure 5.2 for AUC. In Figure 5.1, the MSE values have been multiplied by 1000 to make them easier to read, and the baseline – MSE achieved by assigning the prior probability to all observations – is shown by a dashed horizontal line. Comparing Figures 5.1 and 5.2, it can be seen that while all runs at all ratios for all modelling methods produced a better quality of ranking (AUC) than the baseline – which would achieve an AUC of 0.5 –, this is not true of the accuracy of the values for predicted probability, using MSE as a proxy. In fact for neural network without early stopping, the majority of runs at ratios of less than 16 produced a higher MSE than the baseline, and even at a ratio of 32 two of the runs produced a higher MSE than the baseline.



	Neural Network							NN Early Stopping							Regression						
	1	2	4	8	16	32	All	1	2	4	8	16	32	All	1	2	4	8	16	32	All
Mean	22.39	18.66	17.93	17.62	17.53	17.49	17.44*	17.46	17.43	17.43	17.41*	17.41	17.40	17.40	17.62	17.54	17.50	17.46*	17.45	17.44	17.43
Std Dev	2.89	.914	.285	.099	.065*	.048	.048	.044	.028*	.033	.028	.023	.021	.023	.282	.132	.084	.045	.032*	.022	.019
Avg Iter	177	154	121	112	116	117	118	9.3	10.3	12.1	12.7	16.3	19.4	42.1							

Figure 5.1: 1000 times MSE for the bank home loan data. Each box plot is based on 30 observations (runs). Only a single run for neural network without early stopping at sampling ratio 1 is shown, as the MSE for the other 29 runs was larger than the maximum value shown on the y-axis. An asterisk next to a "Mean" or "Std Dev" figure indicates a statistically significant improvement (at the 5% level) from ratio 1 until the corresponding ratio, but no further improvement beyond this ratio. "Avg Iter" indicates the average number of conjugate gradient iterations required for convergence or at which training was stopped. The dashed horizontal line indicates the value that would be obtained by allocating the prior probability to every case. "+x" indicates x outliers beyond the axis range shown.

Even regression produced a higher MSE than the baseline for some runs at low sampling ratios, though not to the extent of the unstopped neural network. It is explored below whether or not the poor performance on MSE for these runs was due to overfitting of the training data.



	Neural Network							NN Early Stopping							Regression						
Mean	.638	.650	.660	.670	.675	.680	.686*	.681	.685	.687	.689	.691	.692*	.693	.676	.679*	.680	.681	.681	.681	.682
Std Dev	.011*	.013	.009	.009	.009	.007	.009	.006*	.007	.007	.007	.007	.006	.008	.007*	.007	.005	.005	.006	.005	.005
Avg Iter	177	154	121	112	116	117	118	9.3	10.3	12.1	12.7	16.3	19.4	42.1							

Figure 5.2: Area under the ROC curve for the bank home loan data.

Levene's test showed a significant reduction in the variance of MSE to a ratio of 16 for neural network without early stopping and regression, and to a ratio of 2 for neural network with early stopping. Welch's ANOVA showed a significant reduction in mean to a ratio of 'full data set' for neural network without early stopping, and to a ratio of about 8 for neural network with early stopping and regression. Note that the mean for neural network with early stopping was significantly lower than that for the other methods at all sampling ratios.

On the AUC criterion there was no significant reduction in variance with increasing sampling ratio for any of the modelling methods, although there was a significant increase in mean to a ratio of 'full data set' for neural network without early stopping, to a ratio of 32 for neural network with early stopping – though the increase from ratio 8 to ratio 32 was only marginally significant – and to a ratio of 2 for regression.

We now explore whether or not the poor values on test set MSE for some of the runs were due to overfitting of the training data. Considering first neural network without early stopping, two observations support the conclusion that these poor values were due to overfitting:

a) The runs giving larger MSE also tended to give larger maximum values for predicted probability of the rare class on the test set than those giving smaller MSE.
b) The average number of conjugate gradient iterations needed for convergence decreased with increasing sampling ratio, and the largest values for test MSE were produced at low $N_0 : N_1$ ratios. Specifically, 177.5, 154.0, 120.8, 112.1, 116.0, 117.0 and 118.3 iterations were needed on average at ratios 1, 2, 4, 8, 16, 32 and 'full data set' respectively. This suggests that at low sampling ratios the neural network tended to descend 'deeper' into minima of the objective function (deviance) on the training data, or fit the training data harder, to a degree not supported by the test data.

Considering regression, three observations support the conclusion that the poor values on test set MSE for some runs at low sampling ratios were due to overfitting:

a) The runs giving larger MSE also tended to give larger maximum values for predicted probability of the rare class on the test set than those giving smaller MSE.
b) Both a Pearson and a Spearman correlation analysis showed a significant positive correlation between the number of variables included in the regression model and MSE for sampling ratios up to 4. Specifically, the Spearman rank correlation coefficients between number of inputs and MSE were 0.52, 0.42, 0.51, 0.30, 0.14, -0.18 and -0.37 at ratios 1, 2, 4, 8, 16, 32 and 'full data set' respectively.
c) Further examination showed that for this data set two inputs, one continuous and one categorical with two categories, were far stronger predictors of the target than the other inputs. The models were refitted with just these two inputs, and the performance at a ratio of 1 in terms of both mean and variance of MSE for both regression and neural network without early stopping were significantly better than when all inputs were used. This was not the case for neural network with early stopping.

Figure 5.1 also shows that on this data set, early stopping did a very good job of restricting the neural networks from overfitting, even at low sampling ratios, and the variance of MSE for neural network with early stopping was significantly lower than even that of regression at sampling ratios up to 8. The validation data set for each run contained 963 positive observations in 33 continuous dimensions. If we use the ratio of number of positives to number of inputs as a rough indication of the density of positives in the input space, this data set would seem to be far sparser than the 90 positive observations in two dimensions in each of the scenarios in Chapter 4. This would tend to make the distances between the positive training cases larger in the input space, and increase the likelihood of a flexible model falsely ‘finding’ small regions of high probability of the rare class. Thus the result that neural network with early stopping did not result in significant overfitting even at low sampling ratios may be surprising and needs some explanation.

A principal components analysis of the 33 continuous inputs revealed that the first three principal components account for 33% and the first seven for 51% of the variance. Hence the effective dimensionality of the input space is somewhat lower than the number of inputs – despite the fact that preliminary screening of the inputs was performed to ensure that no two inputs that were individually highly correlated and had similar practical meaning were both used – and the effective density of the positives is correspondingly higher. The effect of the density of positives in the input space is examined in the analysis of the KDD Cup 1998 data set in the next section.

The number of iterations at which training was stopped was also examined. The average number of iterations per run, including preliminary training, was 9.3 at ratio 1, 10.3 at ratio 2, 12.1 at ratio 4, 12.7 at ratio 8, 16.3 at ratio 16, 19.4 at ratio 32 and 42.1 when the ‘full data set’ was used. Thus for neural network with early stopping, the use of the validation data set effectively limited overfitting by allowing fewer conjugate gradient iterations at lower sampling ratios. It should also be noted that, despite the seemingly low number of iterations at which training was stopped, significant underfitting did not result. Significant underfitting would lead to MSE values close to the baseline and AUC values close to 0.5.

In the light of the results on the artificial data sets in Chapter 4, the relative variances of the inputs for the rare class were compared to those for the common class. Most of the important inputs had variances either more than 1.3 times greater or less than 0.85 times smaller for the rare class than for the common class. Comparison with the results for Chapter 4 indicates that under such conditions the probability contour lines are non-linear, and neural networks with early stopping tend to outperform regression and show less improvement in accuracy of approximation of the underlying class probabilities from higher sampling ratios. Thus the good performance of neural network with early stopping relative to regression even at low sampling ratios would seem to be due to a relatively complex underlying structural relationship between the inputs and the target.

Finally, with regard to the MSE criterion, some discussion is necessary with regard to the magnitude of the improvements with increasing sampling ratio. The best-performing model, neural network without early stopping using the full data set, gave only a 1% improvement in mean when compared to the baseline. Further, the coefficient of variation, even at a ratio of 1, was only 1.60% in the case of regression and 0.25% in the case of neural network with early stopping. Hence we need to consider whether the statistically significant differences noted above are practically significant. In other words, although increasing the sampling ratio clearly made a practical difference with regards to quality of ranking (AUC), did it make a practical difference with regards to the accuracy of estimation of the actual probabilities? The results in all three artificial scenarios examined in Chapter 4, for which the underlying probabilities for the target classes in the test set were known, showed that relatively small percentage improvements in MSE translate into far larger percentage improvements in the accuracy of prediction of the underlying probabilities. As was also shown in Chapter 4, this effect increases with increasing rarity of the rare class. Thus it seems likely that with a prior of 1.79%, as is the case for this data set, even these small percentage differences in MSE are practically important.

In summary, on this data set:

- An increase in sampling ratio did result in a decrease in the variance of performance when evaluated on the criterion more sensitive to overfitting (MSE), but not on that less sensitive to overfitting (AUC).
- However, the decrease in the variance of MSE was only greater for neural network than for regression when no regularisation was used. When early stopping was used, the decrease was greater for regression than for neural network. Further, it was seen that regression could overfit the training data at low sampling ratios.
- Even though there was only a marginal reduction in the variance of performance with increasing sampling ratio when early stopping was used, there was an improvement in the mean of performance up to a ratio of at least 8 with regards to both quality of ranking and estimation of the underlying probabilities of the rare class. Thus justification was found for using a larger sampling ratio, even when neural network with early stopping is the modelling method to be employed.
- Neural network with early stopping outperformed regression, even at low sampling ratios. This is most likely due to a complex underlying probability structure, as evidenced by the unequal variances of the most important inputs for the rare and common classes.

5.2 KDD Cup 1998 Data

This data set was used for the 1998 'Knowledge Discovery in Databases' competition and is available from the UCI Machine Learning Repository (Asuncion and Newman, 2007; ACM Special Interest Group on Knowledge Discovery and Data Mining, 2007). The target is binary 'respond to mailing with gift' with values 1 (respond) and 0 (doesn't respond) for a direct mail campaign to lapsed donors to a charity.

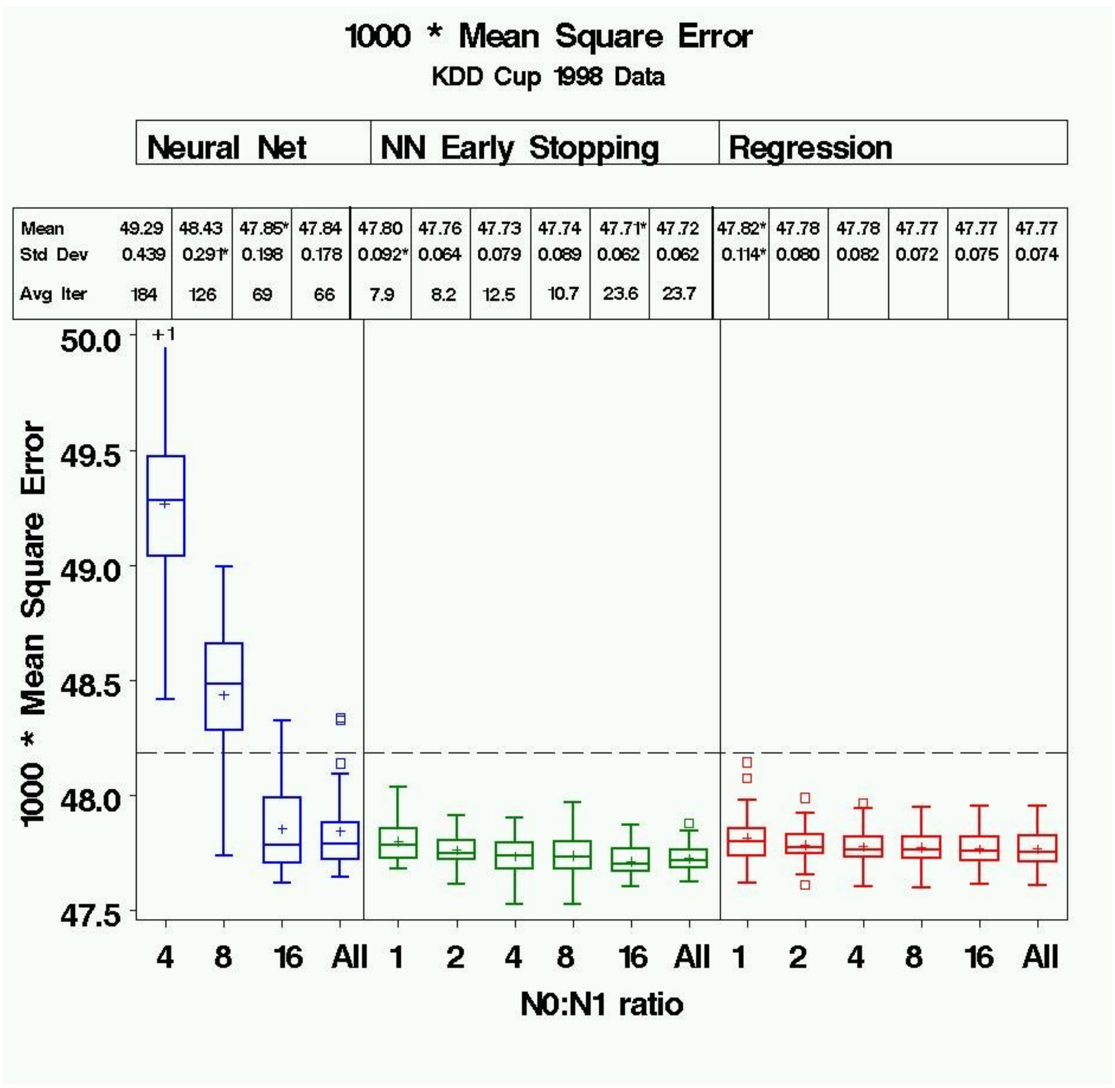
For this data set, 4843 / 95412 = 5.08% of cases had response = 1, i.e. the prior was 0.0508. The sampling ratios considered were 1, 2, 4, 8, 16 and 'full data set', where 'full data set' corresponds to a ratio of 18.7.

The actual KDD98 data set has 479 potential inputs. Input pre-selection was largely based on those inputs used by Georges and Milley (2000) in their paper about their competition entry. Note that the actual competition task was to predict amount of gift, and not just response, and Georges and Milley built a 2-phase model, with phase 2 modelling $\Pr(response = 1)$ and phase 1 modelling $E(gift\_amount / response = 1)$. I have based my input selection on the inputs they used for both phases. Fifteen continuous inputs were pre-selected. These are listed in Appendix A.

Twenty hidden units were used for the neural networks, which is also the number used by Georges and Milley (2000).

Figure 5.3 shows the box-and-whisker plot for MSE and Figure 5.4 for AUC. In Figure 5.3, the MSE values have been multiplied by 1000 to make them easier to read. As was the case with the bank home loan data set, although all runs for all modelling methods produced a better quality of ranking (AUC) than the baseline, this was not the case for accuracy of estimation of the underlying class probabilities, using MSE as a proxy.

For regression, there was no improvement in either the mean or the variance of performance with increasing sampling ratio on either the MSE or the AUC criterion. For neural network with early stopping, there was no improvement in variance on either criterion, or in mean on the AUC criterion, but there was a marginally significant improvement in mean on the MSE criterion when ratios of 16 and 'full data set' were used. For neural network without early stopping, there was a significant reduction in variance to a ratio of 8 and in mean to a ratio of

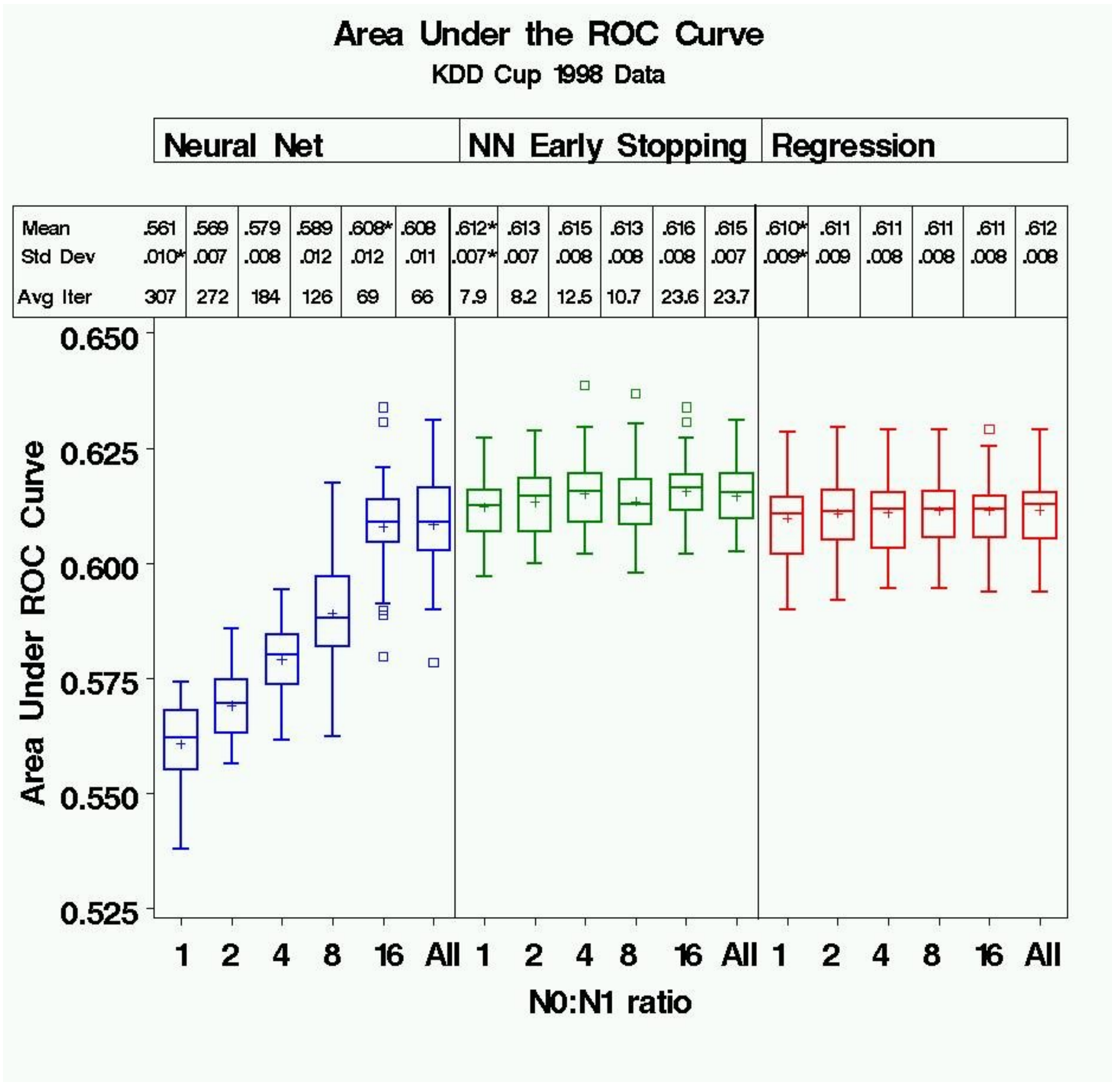


Mean	49.29	48.43	47.85*	47.84	47.80	47.76	47.73	47.74	47.71*	47.72	47.82*	47.78	47.78	47.77	47.77	47.77
Std Dev	0.439	0.291*	0.198	0.178	0.092*	0.064	0.079	0.089	0.062	0.062	0.114*	0.080	0.082	0.072	0.075	0.074
Avg Iter	184	126	69	66	7.9	8.2	12.5	10.7	23.6	23.7						

Figure 5.3: 1000 times MSE for the KDD Cup 1998 data. All runs for neural network without early stopping at ratios 1 and 2 produced values larger than the maximum value shown on the y-axis.

16 on the MSE criterion. There was no reduction in variance with increasing sampling ratio on the AUC criterion, but there was an improvement in mean to a ratio of 16. As with the bank home loan data set, it was observed that the runs producing large values for MSE tended to have larger maximum values for predicted probability than the other runs, and a decreasing number of conjugate gradient iterations was required to obtain convergence with increasing sampling ratio. Both of these observations suggest that the large MSE values at low sampling ratios were the result of overfitting of the training data.

From Figures 5.3 and 5.4 we can see that not even the best models fitted performed particularly well on this data set. The maximum average AUC obtained was 61.6% – achieved for neural network with early stopping at ratio 16 – compared to a baseline of 50%. An AUC



	Neural Net						NN Early Stopping						Regression					
	1	2	4	8	16	All	1	2	4	8	16	All	1	2	4	8	16	All
Mean	.561	.569	.579	.589	.608*	.608	.612*	.613	.615	.613	.616	.615	.610*	.611	.611	.611	.611	.612
Std Dev	.010*	.007	.008	.012	.012	.011	.007*	.007	.008	.008	.008	.007	.009*	.009	.008	.008	.008	.008
Avg Iter	307	272	184	126	69	66	7.9	8.2	12.5	10.7	23.6	23.7						

Figure 5.4: Area under the ROC curve for the KDD Cup 1998 data.

of 61.6% corresponds to a Gini index of 23.2%, and would generally be considered a poor model in terms of ranking quality. A minimum average MSE of 0.04771 for the best model represents only a 1% improvement on the baseline of 0.04818. As explained in Section 5.1, this represents a more marginal improvement in approximation of the underlying probabilities than the 1% improvement shown on the bank home loan data set, as the prior is almost three times greater for the KDD Cup data set than for the bank data set. Thus the data are noisy, with no strong structural relationship between the inputs and the target. Neural network without early stopping tended to model the noise in the data at low sampling ratios. This was not the case for either neural network with early stopping or regression. In order to explain this, both the density of the positives in the input space and the structure of underlying relationship between the inputs and the target were investigated.

The KDD Cup 1998 data set, with 4843 positives and 15 inputs, has the highest density of positives in the input space of the real data sets considered. A principal components analysis showed that the first three principal components account for 53% and the first seven for 82% of the variance, hence the effective dimensionality is considerably lower than fifteen. This makes the effective density of the positives in the input space even higher. At higher densities the asymptotic properties of maximum likelihood estimates listed in Chapter 2 become more relevant, regardless of sampling ratio.

In order to investigate whether the density of positives in both the training and validation data sets enabled both neural network with early stopping and regression to perform relatively well at low sampling ratios, all runs were refitted using only 10% of the positive observations in the training set, 9% in the validation data and 81% in the test set for each run, as opposed to the 40% training, 30% validation and 30% test partitioning used for each run in the initial modelling.

When the smaller training and validation samples were used, an increase in sampling ratio resulted in an improvement in performance when evaluated on the MSE criterion. The variance of MSE reduced by a factor of approximately three from a ratio of 1 to a ratio of 16 for both neural network with early stopping and regression. There was also a significant reduction in mean for both methods. Thus it does seem that the higher density of positives in the data was at least partially responsible for the relatively good performance of both modelling methods at low sampling ratios in the initial fitting.

The structure of underlying relationship between the inputs and the target was investigated by examining the variances of the inputs for the rare class relative to those for the common class. The majority of the fifteen inputs had similar variances for both classes– eleven of the fifteen inputs had a ratio of variance for the rare class to variance for the common class between 0.88 and 1.21. Thus, even though the distributions of all the inputs are certainly not normal, there may be some similarity to the 'equal variances for both classes' scenario of Chapter 4, for which the contour lines for the probability of class 1 are linear and the regression performs relatively well at all sampling ratios.

In order to further examine why regression did not overfit the data at low sampling ratios, the number of inputs included in the final model, and its correlation with MSE were calculated. For all sampling ratios, the number of inputs included was in a narrow range around seven, and there was no significant correlation between number of inputs fitted and MSE. We can also note that the standard deviation of the most important inputs – myngift, HV2, lastdays, fistdays and income (see Appendix A) – all decreased by a factor of less than two from ratio 1 to ratio 'full data set'. It seems likely that the relatively high density of the positives tended to result in only relevant inputs being selected, and their regression coefficients being relatively stable.

As in the case of the bank home loan data set, the lack of overfitting for neural network with early stopping can be explained by the observation that the average number of conjugate gradient iterations at which training was stopped tended to increase with increasing $N_0 : N_1$ ratio, increasing from 7.9 at a ratio of 1 to 23.7 at a ratio of 'full data set'.

In summary, on this data set:

- The variance and the mean of model performance on the evaluation criterion more sensitive to overfitting (MSE) improved with increasing sampling ratio for neural network without early stopping, and overfitting occurred at low sampling ratios. Only the mean and not the variance of performance improved with increasing sampling ratio when evaluated on the criterion less sensitive to overfitting (AUC).
- For neural network with early stopping and regression, an increase in sampling ratio did not result in a meaningful improvement in either the variance or the mean of performance, regardless of evaluation criterion. Further, the two methods gave very similar results. This is most likely due to a combination of several factors: a relatively simple but weak structural relationship between the inputs and the target and a relatively high density of the positives in the input space.

5.3 Insurance Company Benchmark (COIL 2000) Challenge Data Set

This data set is available from the UCI Machine Learning Repository. The target is binary 'ownership of a caravan insurance policy' for customers of an insurance company, with outcome 1 "has a caravan policy" and 0 "doesn't have a caravan policy".

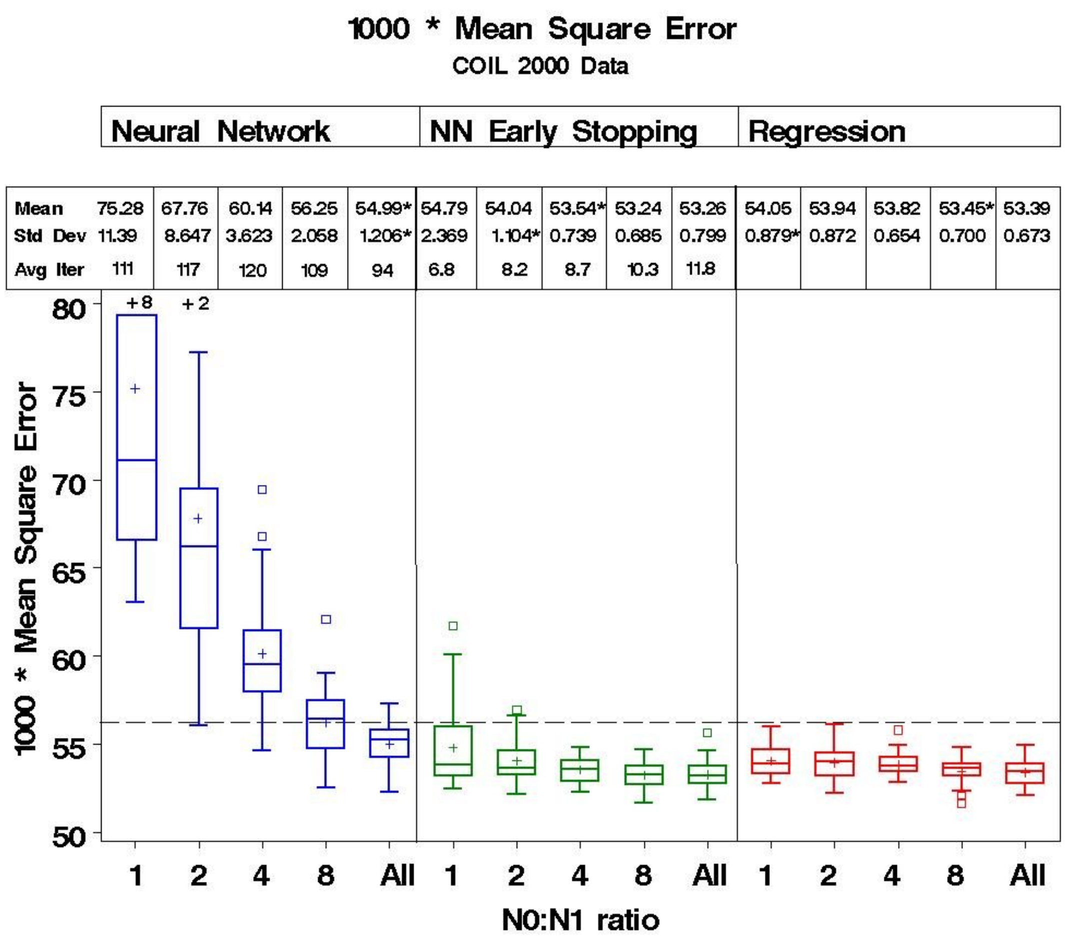
For this data set, 586 / 9822 = 5.97% of cases had response = 1, i.e. the prior was 0.0597. The sampling ratios considered were 1, 2, 4, 8 and 'full data set', where 'full data set' corresponds to a ratio of 15.8. Of the original 86 inputs, 18 were pre-selected for use, based on those used by the competition winner (Elkan, 2000) as well as those used consistently for splitting nodes in class probability trees. These are listed in Appendix A. Three hidden units were used for the neural networks.

Figure 5.5 shows the box-and-whisker plot for MSE and Figure 5.6 for AUC. In Figure 5.5, the MSE values have been multiplied by 1000 to make them easier to read. As was the case with the other real data sets, although all runs for all modelling methods produced a better quality of ranking (AUC) than the baseline, this was not the case for the accuracy of estimation of the underlying class probabilities, using MSE as a proxy.

On the AUC criterion, the variance did not decrease with increasing sampling ratio for any of the modelling methods. For neural network without early stopping, the mean improved to a ratio of ‘full data set’. For regression, the mean improved to a ratio of 8. For neural network with early stopping, the mean did not improve with increasing sampling ratio.

On the MSE criterion, both the variance and the mean for neural network without early stopping decreased with increasing sampling ratio up to a ratio of ‘full data set’. The majority of runs at low sampling ratios performed worse than the baseline – simply predicting the prior for every observation – and as with the other data sets this proved to be due to overfitting. For neural network with early stopping, the variance decreased to a ratio of 2 and the mean decreased to a ratio of 4. There were seven runs at a ratio of 1 and two runs at a ratio of 2 that performed worse than the baseline. Examination of the fitted probabilities showed that this was also due to overfitting.

For regression, the mean of MSE decreased to a ratio of 8. The variance, however, did not decrease with increasing sampling ratio. To understand this, it is enlightening to consider the nature of the inputs. The inputs differ from all the other data sets considered in that, apart from one derived input, TOTNPOLS, they all take on integer values in the range 0 to 9, with policy contribution amounts being converted to a log-like scale. Although they could be treated as ordinal, they were treated as continuous for the purposes of the analysis. The narrow and discrete range of input values appears to have helped to make the regression coefficients at low sampling ratios more stable than for the other data sets. Indeed only a small reduction in the standard deviations of the regression coefficients with increasing sampling ratio was found for the most important inputs: car insurance premium, purchasing power class and fire policy premium.



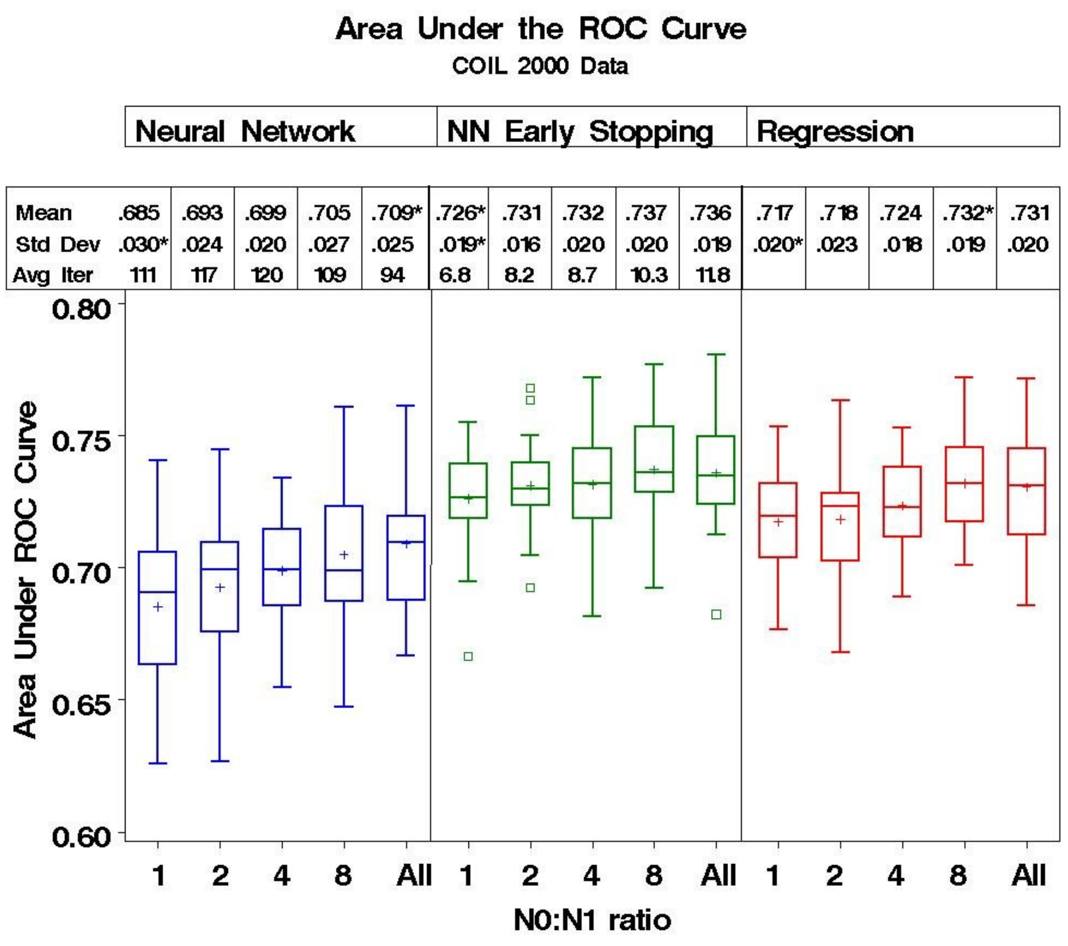
	Neural Network					NN Early Stopping					Regression				
	1	2	4	8	All	1	2	4	8	All	1	2	4	8	All
Mean	75.28	67.76	60.14	56.25	54.99*	54.79	54.04	53.54*	53.24	53.26	54.05	53.94	53.82	53.45*	53.39
Std Dev	11.39	8.647	3.623	2.058	1.206*	2.369	1.104*	0.739	0.685	0.799	0.879*	0.872	0.654	0.700	0.673
Avg Iter	111	117	120	109	94	6.8	8.2	8.7	10.3	11.8					

Figure 5.5: 1000 times MSE for the COIL 2000 data.

Further, all three of these inputs had very similar variances for the rare and common classes. As with the KDD Cup 1998 data set, there may be some similarity to the 'equal variances for both classes' scenario of Chapter 4, for which the contour lines for probability of class 1 are linear and the regression performs relatively well at all sampling ratios. This may explain why neural network with early stopping did not outperform regression at any sampling ratio.

In summary, on this data set:

- Increase in sampling ratio did result in a decrease in the variance of performance for the neural networks when evaluated on the criterion more sensitive to overfitting (MSE). This decrease was far greater when no regularisation was used than when early stopping was used. Increase in sampling ratio did not result in a decrease in the variance of performance for regression.
- Increase in sampling ratio did not result in a decrease in variance of performance for either the neural networks or regression when evaluated on the criterion less sensitive to overfitting (AUC).



	Neural Network					NN Early Stopping					Regression				
	1	2	4	8	All	1	2	4	8	All	1	2	4	8	All
Mean	.685	.693	.699	.705	.709*	.726*	.731	.732	.737	.736	.717	.718	.724	.732*	.731
Std Dev	.030*	.024	.020	.027	.025	.019*	.016	.020	.020	.019	.020*	.023	.018	.019	.020
Avg Iter	111	117	120	109	94	6.8	8.2	8.7	10.3	11.8					

Figure 5.6: Area under the ROC curve for the COIL 2000 data.

- For all modelling methods there was evidence of a decrease in the mean of MSE with increasing sampling ratio. Thus justification was provided for using a larger sampling ratio, even when regression is to be used.
- Neural network with early stopping did not outperform regression. This is likely due to a fairly linear relationship between the most important inputs and the target.

5.4 Summary

An increase in the sampling ratio did not result in deterioration in either the variance or mean of performance for any of the data sets, modelling methods or evaluation criteria used. Thus no disadvantage was found, other than increase in training time and data storage requirements, in using larger sampling ratios, or indeed the ‘full data set’, for training.

However, the improvements were not always statistically or practically significant, as is described in more detail below.

On all three real data sets studied, it was found that, without regularisation, neural network did indeed achieve a highly significant improvement in both the variance and the mean of performance with increasing sampling ratio when evaluated on the criterion more sensitive to overfitting (MSE), and an improvement in the mean of performance when evaluated on the criterion less sensitive to overfitting (AUC). Further, these improvements were far larger than for forward stepwise regression. For both methods the improvements in the variance of MSE were far larger than those in the variance of AUC, and the improvements in the mean of MSE were far larger, relative to the baseline, than those in the mean of AUC. Thus when comparing the unregularised flexible modelling method to the inflexible modelling method, all aspects of the hypotheses were confirmed on these data sets.

However, when the neural networks were regularised by the use of a separate validation data set – early stopping – the results were not so clear, and various factors, including density of the positives in the space spanned by the inputs and the underlying structure and strength of the relationship between the inputs and the target appeared to play a role in determining the effect of increasing sampling ratio.

For the data set with greatest density of positives in the input space – KDD Cup 1998 – there was no practically significant improvement in either mean or variance on either criterion with increasing sampling ratio for either neural network with early stopping or regression. It was shown, however, that increasing sampling ratio did have a beneficial effect on both mean and variance for both methods when the density of positives in the training and validation data was artificially reduced.

On the other two data sets studied, there was some reduction in the variance of MSE with increasing sampling ratio. In the case of the COIL 2000 data set, this reduction was greater for neural network with early stopping than for regression, and in fact the reduction in variance for regression was not statistically significant. However, in the case of the bank home loan data set, the reduction in variance was greater for regression than for neural network with early stopping. It was seen that regression could produce overfitting at low sampling ratios if the number of inputs available for potential inclusion is large.

When early stopping was used, the average number of conjugate gradient iterations at which training was stopped increased with increasing sampling ratio on all three data sets. At low sampling ratios, training was stopped early in the process (less than 10 iterations on average at a ratio of 1). This put a very effective brake on the potential for overfitting of neural networks.

On all three data sets studied, in all cases where an improvement in the variance of performance with increasing sampling ratio was found, this reduction was greater on the evaluation criterion more sensitive to overfitting (MSE) than on the criterion less sensitive to overfitting (AUC). The same result held for the improvement in the mean of performance relative to the baseline. These results are in line with the stated hypotheses.

6 Conclusions and Further Work

6.1 Introduction

The hypotheses tested state broadly that the common practice of under-sampling the common class so as to increase the ratio of common to rare class cases in the training data has a negative effect on the performance of models, particularly flexible models when evaluated on criteria sensitive to overfitting.

An increase in the ratio of common to rare class cases (sampling ratio) in the training data did not result in a deterioration in either the variance or the mean of performance for any of the data sets, modelling methods or evaluation criteria used. Thus no disadvantage was found, other than increase in training time and data storage requirements, in using larger sampling ratios for model training. However, the improvements in performance with increasing ratio were not always statistically or practically significant, and were of differing degree for the different modelling methods and evaluation criteria tested, as is described in more detail below.

6.2 Sufficiency of AUC and MSE as Evaluation Criteria

The results on the two evaluation criteria measuring quality of ranking, Area Under the Receiver Operating Characteristic curve (AUC) and proportion of actual positives in the top 10% of predicted probabilities, were very similar for all three artificial data sets studied, and further these two criteria were highly correlated. This also applied to the two evaluation criteria estimating accuracy of approximation of the predicted probabilities to the true probabilities of the rare and common classes, mean square error (MSE) and mean square deviance residual. On the artificial data sets it was possible to determine analytically the probability of the rare class for each observation in each test set. It was found that MSE was more highly correlated with the true accuracy of the predicted probabilities, calculated as mean squared difference between actual and predicted probability of the rare class (MSD), than mean square deviance residual. AUC is a widely recognised measure of model performance, particularly in the machine learning community. Hence only AUC and MSE were chosen for evaluation of the real-world data sets in Chapter 5, and only the results on these two criteria are discussed below.

6.3 Unstopped Neural Network Compared to Logistic Regression

When not stopped by the use of a regularisation technique, the flexible modelling method used, multi-layer perceptron neural network, did show highly significant improvement in both the variance and the mean of accuracy of estimation of the underlying class probabilities with increasing sampling ratio for all of the data sets studied. Further, these improvements were greater than those for the inflexible method, logistic regression, for all the data sets except the artificial data set with localised region of high probability of the rare class, for which the improvements were roughly equal. The neural networks showed a clear tendency to overfit the training data at low sampling ratios by producing more extreme values of predicted probability than at higher ratios; in particular producing predicted probabilities of the rare class closer to one than warranted on unseen data in regions of the input space in which positive cases occurred in the training data. This tendency reduced dramatically with increasing sampling ratio, particularly on the real-world data sets studied. It was shown for the real data sets that a greater number of iterations of conjugate gradient descent were required on average for convergence at lower sampling ratios than at higher ratios – an indication that the algorithm was 'digging deeper' into the minima of the objective function (deviance) on the training sets at lower sampling ratios, to a degree not supported by the test sets.

The unstopped neural networks also showed an improvement in mean quality of ranking, as measured by AUC, with increasing sampling ratio for all of the real data sets studied. For all three real data sets, the mean of AUC increased with increasing sampling ratio for all ratios up to use of the full training partition. For the COIL 2000 data set, though, the increase from ratio 4 to full training partition was only nearly significant (p=0.08). Thus although, as discussed below, there was no decrease in the variance of ranking quality, benefit was found in using a higher sampling ratio even when performance was measured on the criterion less sensitive to overfitting. For all data sets, the improvement in mean AUC with increasing sampling ratio for logistic regression, if any, was far smaller than for unstopped neural network.

Thus, when comparing unstopped flexible modelling method to the inflexible method, all aspects of the hypotheses tested were confirmed on the data sets studied. Further, the results obtained are in line with the results of the majority of empirical studies using neural networks discussed in the literature review.

6.4 The Effect of Early Stopping

When, however, regularisation was applied to the flexible modelling method by optimising performance on a validation data set – early stopping – the results were not so clear, and various factors, including the underlying structure of the relationship between the inputs and the output and the density of the positives in the space spanned by the inputs seemed to play a role in determining the effect of increasing sampling ratio on model performance.

It was shown, using simulated data, that in the case of non-linear contours of probability of the rare class, particularly a localised region of high probability in the input space, logistic regression can show greater benefit, in terms of both the variance and the mean of accuracy of prediction of the underlying class probabilities, from a higher sampling ratio than neural networks. This result was also found on one of the real data sets studied (the bank home loan data set). Similarities, with regard to the relative variances of the most important inputs for the rare class compared to the common class, between the bank data set and the artificial data sets mentioned above were noted. Further, it was shown on this real data set that logistic regression is quite capable of overfitting the training data at low sampling ratios when a large number of inputs are available for potential inclusion at each step of the forward selection process. In fact, on this data set, at a sampling ratio of 1:1 30% of runs gave poorer performance on the MSE criterion than the baseline of simply predicting the prior probability for every case. This finding is in line with the arguments of Harrell (2001), mentioned in Chapter 2, that the p-value for inclusion of a term in regression can be very inaccurate if the models being compared were not pre-specified, and over-fitting can result. In contrast, on this data set neural network with early stopping did not overfit the training data to the extent of giving larger MSE than the baseline for any of the runs.

On the artificial data set for which the contours of probability of the rare class were linear, on the other hand, it was shown that neural network with early stopping can show greater benefit from a higher sampling ratio, in terms of both the variance and the mean of the accuracy of prediction of the underlying class probabilities, than logistic regression. This result was also found on one of the real data sets studied (COIL Challenge 2000), and again similarities to the artificial data set mentioned, in terms of the relative variances of the most important inputs for the rare and common classes, were noted.

On the third real data set studied (KDD Cup 1998), no significant improvement with increasing sampling ratio in either mean or variance with regard to either the accuracy of predicted probabilities or quality of ranking for either logistic regression or neural network with early stopping was found. If we use the ratio of number of positive cases in the available data to the effective dimensionality of the inputs as a rough indication of the density of positives in the input space, this data set has by far the highest density of the three real data sets studied. When the number of positives in the training and validation sets was artificially reduced, significant improvements in both mean and variance with regard to accuracy of predicted probabilities (MSE), but not quality of ranking (AUC), were found. In Chapter 2 the theoretical result that the asymptotic properties of maximum likelihood estimates are applicable regardless of sampling ratio was noted. The above discussion indicates that, because of the high density of positives in the original data set, this theoretical result may be applicable to this data set.

On all three of the real-world data sets studied, it was found that the early stopping technique very effectively prevented large-scale overfitting of the training data, even at low sampling ratios. This could be explained by the observation that on all three data sets the average number of iterations of conjugate gradient descent at which stopping occurred increased with increasing sampling ratio. Despite this, significant underfitting was not observed at any sampling ratio.

6.5 The Effect of Evaluation Criterion

On all of the data sets studied, in all cases where a reduction in the variance of performance with increasing sampling ratio was found, this improvement was relatively greater when evaluated on MSE than on AUC. In particular, practically no decrease in the variance of AUC was found with increasing sampling ratio for any of the data sets or modelling methods studied. This is in contrast to the reductions in the variance of MSE described above. The improvements mean of performance, relative to the baseline, with increasing sampling were also greater for MSE than for AUC. Further, for all of the runs for all of the data sets and modelling methods studied, the AUC showed a better performance than the baseline. As discussed above, this was not the case for MSE. These observations confirm the hypothesis that the effect of increasing sampling ratio on model performance is greater when performance is evaluated on criteria more sensitive to overfitting than on criteria less sensitive to overfitting.

6.6 Summary

To summarise the indications of the experimental results with regard to the hypotheses tested:

In a practical context, for which the number of rare class cases available for modelling is limited:

1. The common practice of under-sampling the common class increases the variability of model performance, worsens the mean of model performance, or both, when applied to unseen data, compared to training using all available cases of both classes.
   - This was confirmed by the experimental results.

   These effect(s) are more severe:

   a) for flexible modelling methods than for inflexible methods.
      - This was found to be dependent on other factors such as the structure of the underlying relationship between the inputs and the target, and whether or not a regularisation technique was used for the flexible method.

   b) for lower sampling ratios in the training sample than for higher ratios.
      - This was confirmed by the experimental results, although the level of statistical significance was dependent on other factors.

2. The effect(s) hypothesised in 1. above are greater when model performance is evaluated using criteria that are more sensitive to overfitting than criteria less sensitive to overfitting.
   - This was confirmed by the experimental results.

6.7 Further Work

The factors influencing the relationship between model performance and sampling ratio tested in this study were flexibility of modelling method and sensitivity to overfitting of the evaluation criterion on which performance is measured. Further factors that could be tested are: absolute number of rare class cases available for training, rarity of the rare class (prior probability) and estimated density of positives in the input space. Although some observations relating to these factors were made in reporting the results, they were not tested in this study.

This study was concerned only with the common practice of decreasing the sampling ratio in the training set by simple under-sampling of the common class. It would be interesting to investigate the effect of altering the sampling ratio by means of the other over- and under-sampling techniques described in Chapter 2.

References

ACM Special Interest Group on Knowledge Discovery and Data Mining (2007): *KDD Cup Databases* [kdd.ics.uci.edu/databases/]. Accessed 15 April 2008.

Asuncion, A. and Newman, D.J. (2007): *UCI Machine Learning Repository* [http://www.ics.uci.edu/~mlearn/MLRepository.html]. University of California, School of Information and Computer Science, Irvine, CA. Accessed 15 April 2008.

Barandela, R., Valdovinos, R. M., Sanchez, J. S. and Ferri, F. J. (2004): The Imbalanced Training Sample Problem: Under or Over Sampling? *LNCS*, 3138, pp. 806-814.

Bradley, A. (1997): The Use of the Area Under The ROC Curve in the Evaluation of Machine Learning Algorithms. *Pattern Recognition*, 30(7), pp. 1145-1159.

Brause, R., Langsdorf, T. and Hepp, M. (1999): Neural Data Mining for Credit Card Fraud Detection. In: *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, IEEE Computer Society Press, Silver Spring, MD, pp. 103-106.

Carroll, R.J. and Ruppert, D. (1998): *Transformation and Weighting in Regression*. Chapman and Hall, New York.

Chan, P.K.and Stolfo, S.J. (1998): Toward Scalable Learning with Non-Uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection. In: *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 164-168.

Chaudhuri, A. and Stenger, H. (2005): *Survey Sampling: Theory and Met*hods, 2nd edition. Chapman and Hall, Boca Raton.

Chawla, N.V. (2003): C4.5 and Imbalanced Data Sets: Investigating the Effect of Sampling Method, Probabilistic Estimate, and Decision Tree Structure. In: *Workshop on Learning from Imbalanced Datasets II*, International Conference on Machine Learning.

Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P (2002): SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, pp. 321-357.

Chawla, N.V., Japkowicz, N. and Kolcz, A. (2004): Editorial: Special Issue on Learning from Imbalanced Data Sets. *SIGKDD Explorations* 6(1), pp. 1-6.

Cochran, W.G (1977): *Sampling Techniques*. John Wiley and Sons, New York.

CoIL Challenge 2000: The Insurance Company Case. Data available at http://www.liacs.nl/~putten/library/cc2000/. Accessed 15 April 2008.

Cui, G., Wong, M.L., Zhang, G. and Li, L. (2008): Model Selection for Direct Marketing: Performance Criteria and Validation Methods. *Marketing Intelligence & Planning*, 26(3), pp. 275-292.

Drummond, C. and Holte, R.C. (2003): C4.5, Class Imbalance, and Cost Sensitivity: Why Under-Sampling Beats Over-Sampling. In: *Workshop on Learning from Imbalanced Data Sets II*, International Conference on Machine Learning.

Drummond, C. and Holte, R.C. (2005): Severe Class Imbalance: Why Better Algorithms Aren't the Answer. *Proceedings of the 16th European Conference on Machine Learning (ECML/PKDD'05)*, pp. 539-546.

Elazmeh, W., Japkowicz, N. and Matwin, S. (2006): Evaluating Misclassifications in Imbalanced Data. *Lecture Notes in Computer Science*, 4212, pp. 126–137.

Elkan, C. (2000): COIL Challenge 2000 Entry. Available at http://www.liacs.nl/~putten/library/cc2000/report2.html. Accessed 20 June 2008.

Estabrooks, A., Jo, T. and Japkowicz, N. (2004): A Multiple Resampling Method for Learning from Imbalances Data Sets. *Computational Intelligence*, 20(1), pp. 18-36.

Fayyad, U. M. (1996): Data Mining and Knowledge Discovery: Making Sense Out of Data. *IEEE Expert* 11(5), pp. 20-25.

Fletcher, R (1987): *Practical Methods of Optimization*. John Wiley & Sons, New York.

Friedman, J.H. (1997): On Bias, Variance, 0/1 - Loss, and the Curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1), pp. 55-77.

Georges, J. (2001): *Predictive Modeling Using Enterprise Miner Course Notes*. SAS Institute Inc., Cary, NC, USA.

Georges, J. (2004): *Data Preparation for Data Mining Using SAS® Software Course Notes*. SAS Institute Inc., Cary, NC, USA.

Georges, J. and Milley, A.H. (2000): KDD'99 Competition: Knowledge Discovery Contest. *SIGKDD Explorations*, 1(2), pp. 79-84.

Geisser, S. (1975): The Predictive Sample Reuse Method with Applications. *Journal of the American Statistical Association*, 70(350), pp. 320–328.

Grzymala-Busse, J.W., Zheng, Z., Goodwin, L.K. and Grzymala-Busse, W.J. (2000): An Approach to Imbalanced Data Sets Based on Changing Rule Strength. In: *Learning from Imbalanced Data Sets: Papers from the AAAI Workshop*, AAAI Press Technical Report WS-00-05, pp. 69-74.

Han, H., Wang, W. Y. and Mao, B. H. (2005): Borderlinesmote: A New Over-Sampling Method in Imbalanced Data Sets Learning. In: *International Conference on Intelligent Computing (ICIC'05)*, Lecture Notes in Computer Science, 3644, Springer-Verlag , pp. 878-887.

Hand, D. J. (2005): Good Practice in Retail Credit Scorecard Assessment. *Journal of the Operational Research Society*, 56, pp. 1109-1117.

Hanley J.A. and McNeil B.J. (1982): The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve. *Radiology* 143, pp. 29-36.

Harrell, F. E. Jr. (2001): *Regression Modeling Strategies: with Applications to Linear Models, Logistic Regression, and Survival Analysis.* Springer–Verlag, New York.

Hassibi, K. (2000): Detecting Payment Card Fraud with Neural Networks. In: Lisboa, P. J. G., Vellido, A. and Edisbury, B. (eds.) *Business Applications of Neural Networks*. World Scientific, Singapore, pp.141-158.

Hastie, T., Tibshirani, R. and Friedman, J. (2001): *The Elements of Statistical Learning*. Springer–Verlag, New York.

Hsieh, D.A., Manski, C.F. and McFadden, D. (1985): Estimation of Response Probabilities from Augmented Retrospective Observations. *Journal of the American Statistical Association*, 80(391), pp. 651-662.

Japkowicz, N. and Stephen, S. (2002): The Class Imbalance Problem: A Systematic Study. *Intelligent Data Analysis Journal,* 6(5), pp. 429-449.

Khoshgoftaar, T. M., Seiffert, C., Hulse, J. V., Napolitano, A. and Folleco, A. (2007): Learning with Limited Minority Class Data. In: *Proceedings of the Sixth international Conference on Machine Learning and Applications*. IEEE Computer Society, Washington, DC, pp. 348-353.

King, G. and Zeng, L. (2004): Inference in Case-Control Studies. In: Chow, S. (ed.) *Encyclopedia of Biopharmaceutical Statistics*, 2nd edition. Marcel Dekker, New York, pp. 1-13.

Kotsiantis, S., Kanellopoulos, D. and Pintelas, P. (2006): Handling Imbalanced Datasets: A Review. *GESTS International Transactions on Computer Science and Engineering*, 30(1), pp. 25–36.

Kubat, M., Holte, R.C. and Matwin, S. (1998): Machine Learning for the Detection of Oil Spills in Satellite Radar Images. *Machine Learning*, 30(2) pp. 195-215.

Kubat, M. and Matwin, S. (1997): Addressing the Curse of Imbalanced Training Sets: One-Side Selection. In: *Proceedings of 14th International Conference on Machine Learning ICML 97*, pp. 179–186.

Kukar, M. and Kononenko, I. (1998): Cost-sensitive Learning with Neural Networks. *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, pp 445-449.

Laurikkala J. (2001): *Improving Identification of Difficult Small Classes by Balancing Class Distribution*, Technical Report A-2001-2, University of Tampere.

Lawrence, S., Burns, I., Back, A.D., Tsoi, A.C. and Giles, C.L. (1998): Neural Network Classification and Unequal Prior Class Probabilities. In: Orr, G., Muller, K.-R. and Caruana, R. (eds.) *Tricks of the Trade, Lecture Notes in Computer Science State-of-the-Art Surveys,* Springer Verlag, pp. 299-314.

Levene, H. (1960): Robust Tests for the Equality of Variance. In: Olkin, I. (ed.), *Contributions to Probability and Statistics*, Stanford University Press, Palo Alto, CA, pp. 278-292.

Ling, C.X., Huang, J. and Zhang, H. (2003): AUC: a Statistically Consistent and more Discriminating Measure than Accuracy. *Proceedings of IJCAI.*

Ling, C.X. and Li, C. (1998): Data Mining for Direct Marketing Problems and Solutions. In: *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 73-79.

Mahadevan, A., Ponnudurai, K., Kersten, G. and Thomas, R. (2002): Knowledge Discovery in Databases and Decision Support Book Series. In: *Decision Support Systems for Sustainable Development*, pp. 343-368.

Manski, C.F. and Lerman, S. (1977): The Estimation of Choice Probabilities from Choice-Based Samples. *Econometrica*, 45, pp. 1977-1988.

Mazurowski, M.A., Habas, P.A., Zurada, J.M., Baker, J.A. and Tourassi, G.D. (2008): Training Neural Network Classifiers for Medical Decision Making: The Effects of Imbalanced Datasets on Classification Performance. *Neural Networks*, 21, pp. 427-436.

McCarthy, K., Zabar, B. and Weiss, G. (2005): Does Cost-sensitive Learning Beat Sampling for Classifying Rare Classes? In: *Proceedings of the 1st international workshop on Utility-based Data Mining*, ACM, New York, pp 69-77.

McQuarrie, A.D.R. and Tsai, C.-L. (1998): *Regression and Time Series Model Selection*. World Scientific Press, Singapore.

Nickerson, A., Japkowicz, N. and Milios, E. (2001): Using Unsupervised Learning to Guide Re-sampling in Imbalanced Data Sets. In: *Proceedings of the Eighth International Workshop on AI and Statistics*, pp. 261–265.

Peduzzi, P.N., Concato, J., Kemper, E., Holford, T.R. and Feinstein, A.R. (1996): A Simulation Study of the Number of Events per Variable in Logistic Regression Analysis. *Journal of Clinical Epidemiology*, 49, pp. 1373-1379.

Potts W.J.E. (1998): *Data Mining Primer: Overview of Applications and Methods*. SAS Institute Inc., Cary, NC.

Potts, W.J.E. (2000): *Neural Network Modeling Course Notes*. SAS Institute Inc., Cary, NC.

Provost, F. and Fawcett, T. (2001): Robust Classification for Imprecise Environments. *Machine Learning*, 42, pp. 203-231.

Provost, F., Fawcett, T. and Kohavi, R. (1998): The Case Against Accuracy Estimation for Comparing Induction Algorithms. In: *Proc. Fifteenth Intl. Conf. Machine Learning*, Madison, WI, pp. 445-453.

Ripley, B.D. (1996): *Pattern Recognition and Neural Networks*. Cambridge University Press, New York.

Sarle, W.S. (ed.) (1997): *Neural Network FAQ, part 1 of 7: Introduction*. Periodic posting to the Usenet newsgroup comp.ai.neural-nets, URL: ftp://ftp.sas.com/pub/neural/FAQ.html. Accessed 20 June 2008.

Scarpa, B. and Torelli, N. (2004): Selecting the Training Set in Classification Problems with Rare Events. In: Vichi, M., Monari, P., Mignani, S. and Montanari, A. (eds.) *New Developments in Classification and Data Analysis*, Springer-Verlag, pp. 39-46.

Stefanowski, J. and Wilk, S. (2006): Rough Sets for Handling Imbalanced Data: Combining Filtering and Rule-based Classifiers. *Fundamenta Informaticae*, 72(1-3), pp. 379-391.

Stone, M. (1974): Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society*, Series B (Methodological), 36(2), pp. 111–147.

Van den Bosch, A., Weijters, T., van den Herik, H.J. and Daelemans, W. (1997): When Small Disjuncts Abound, try Lazy Learning: A case study. In: *Proceedings of the Seventh Belgian-Dutch Conference on Machine Learning*, pp. 109-118.

Van Hulse, J., Khoshgoftaar, T. M. and Napolitano, A. (2007): Experimental Perspectives on Learning from Imbalanced Data. In: *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, vol. 227. ACM, pp. 935-942.

Weinberg, C.R. and Wacholder, S. (1993): Prospective Analysis of Case-Control Data under General Multiplicative-Intercept Risk Models. *Biometrika*, 80(2), pp. 461-465.

Weiss G.M. and Hirsh H. (1998): Learning to Predict Rare Events in Event Sequences. *Proc. 4th Int. Conf. Knowledge Discovery and Data Mining*. AAAI Press, pp. 359-363.

Weiss, G. and Provost, F. (2003): Learning when Training Data are Costly: The Effect of Class Distribution on Tree Induction. *Journal of Artificial Intelligence Research*, 19, pp. 315-354.

Welch, B.L. (1951): On the Comparison of Several Mean Values: an Alternative Approach. *Biometrika*, 38, pp. 330–336.

Ye, J. (1998): On Measuring and Correcting the Effects of Data Mining and Model Selection. *Journal of the American Statistical Association*, 93(441) pp. 120-131.

Zhou, Z. and Liu, X. (2006): Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem. *IEEE Transactions on Knowledge and Data Engineering*, 18(1), pp. 63-77.

Appendix A: Real Data Sets Used.

1. Bank Home Loan Data Set

Target:
new_H

Interval Inputs:

```
EXPTD_CR_TRNVR_A
AVE_VAL_AUTO_CR_L3M
TOT_NET_TURNOVER
AVG_NET_TRN_L3M
ATM_DPOST_6M_Q
ACNT_AGE
MAX_BALANCE
MINIMUM_BAL_L6M
TOT_DEPOSIT_SAVGS_BAL
ADRB_6M_A
ABB_6M_A_M
CAR_SO_CONF_LIMIT
NET_LENDING_LIMIT
TOTAL_UNSECURED_LIMIT
MNTH_SIN_LAST_LIM_CHN
TOT_SECURED_LENDING NUM_I
CUST_AGE
MNTH_SIN_LST_CRED_OPENED
NUM_L
NUM_M
NUM_S
NUM_T
TOTACCS
DAYS_EXCESS
NO_MTHS_IN_EXCESS_L6M
AVE_REG_DR_COMMIT_L6M
AVG_DBT_TRN_L3M
NUMBER_DEBIT_TRANSACTIONS
TIME_WITH_BANK
FEE_INCOME
SSWCH_FEE_6M_A
DISPOSABLE_INCOME
```

Categorical Inputs

```
ACCOMMODATION_TYPE
CUST_C
GENDER
MARITAL_STATUS
SAL_TYPE
```

2. KDD Cup 1998 Data Set

Target:

TARGET_B	Target Variable: Binary Indicator for Response to **97**NK Mailing

Interval Inputs Used

AGE	Overlay Age
INCOME	HOUSEHOLD INCOME
DMA	DMA Code
NUMPRM12	Number of promotions received in the last 12 months (in terms of calendar months translates into 9603-9702)
CARDPRM12	Number of card promotions received in the last 12 months (in terms of calendar months translates into 9603-9702)
RAMNTALL	Dollar amount of lifetime gifts to date
CARDGIFT	Number of lifetime gifts to card promotion to date
MAXRAMNT	Dollar amount of largest gift to date
LASTGIFT	Dollar amount of most recent gift
AVGGIFT	Average dollar amount of gifts to date
CONTROLN	Control number (unique record identifier)
HV2	Average Home Value in hundreds

```
myramnt = sum(ramnt_3,ramnt_4,...,ramnt_24)
myngift = sum((ramnt_3>0),(ramnt_4>0),...,(ramnt_24>0))
fistdays = number of days between fistdate and 30 June 1997
lastdays = number of days between lastdate and 30 June 1997
```

where:

LASTDATE	Date associated with the most recent gift
FISTDATE	Date of first gift
RAMNT_3	Dollar amount of the gift for 96NK
RAMNT_4	Dollar amount of the gift for 96TK
RAMNT_5	Dollar amount of the gift for 96SK
RAMNT_6	Dollar amount of the gift for 96LL
RAMNT_7	Dollar amount of the gift for 96G1
RAMNT_8	Dollar amount of the gift for 96GK
RAMNT_9	Dollar amount of the gift for 96CC
RAMNT_10	Dollar amount of the gift for 96WL
RAMNT_11	Dollar amount of the gift for 96X1
RAMNT_12	Dollar amount of the gift for 96XK
RAMNT_13	Dollar amount of the gift for 95FS
RAMNT_14	Dollar amount of the gift for 95NK
RAMNT_15	Dollar amount of the gift for 95TK
RAMNT_16	Dollar amount of the gift for 95LL
RAMNT_17	Dollar amount of the gift for 95G1
RAMNT_18	Dollar amount of the gift for 95GK
RAMNT_19	Dollar amount of the gift for 95CC
RAMNT_20	Dollar amount of the gift for 95WL
RAMNT_21	Dollar amount of the gift for 95X1
RAMNT_22	Dollar amount of the gift for 95XK
RAMNT_23	Dollar amount of the gift for 94FS
RAMNT_24	Dollar amount of the gift for 94NK

Note: the learning and evaluation sets cup98LRN.txt and cup98VAL.txt were appended to form the data set used.

3. Insurance Company Challenge (COIL 2000) Data Set

Target:

```
CARAVAN Number of mobile home policies 0 - 1
```

Interval Inputs Used

```
PPERSAUT    Contribution car policies
MKOOPKLA    Purchasing power class
AWAPART     Number of private third party insurance 1 - 12
APLEZIER    Number of boat policies
ABYSTAND    Number of social security insurance policies
PBRAND      Contribution fire policies
AWALAND     Number of third party insurance (agriculture)
AZEILPL     Number of surfboard policies
MAUT1       1 car
MBERARBG    Skilled labourers
MBERBOER    Farmer
MBERMIDD    Middle management
MGODGE      No religion
MHKOOP      Home owners
MINKGEM     Average income
MINKM30     Income < 30.000
MOPLLAAG    Lower level education
TOTNPOLS = sum(   AWAPART, AWABEDR, AWALAND, APERSAUT, ABESAUT,
                  AMOTSCO,
                  AVRAAUT, AAANHANG, ATRACTOR, AWERKT, ABROM, ALEVEN,
                  APERSONG, AGEZONG,
                  AWAOREG, ABRAND, AZEILPL, APLEZIER, AFIETS,
                  AINBOED, ABYSTAND)

where:
AWAPART     Number of private third party insurance 1 - 12
AWABEDR     Number of third party insurance (firms) ...
AWALAND     Number of third party insurance (agriculture)
APERSAUT    Number of car policies
ABESAUT     Number of delivery van policies
AMOTSCO     Number of motorcycle/scooter policies
AVRAAUT     Number of lorry policies
AAANHANG    Number of trailer policies
ATRACTOR    Number of tractor policies
AWERKT      Number of agricultural machines policies
ABROM       Number of moped policies
ALEVEN      Number of life insurances
APERSONG    Number of private accident insurance policies
AGEZONG     Number of family accidents insurance policies
AWAOREG     Number of disability insurance policies
ABRAND      Number of fire policies
AZEILPL     Number of surfboard policies
APLEZIER    Number of boat policies
AFIETS      Number of bicycle policies
AINBOED     Number of property insurance policies
ABYSTAND    Number of social security insurance policies
```

Note: the learning and evaluation sets ticdata2000.txt and ticeval2000.txt (merged with tictgts2000.txt) were appended to form the data set used.

Appendix B: SAS Code to Implement the Experiment

```
/********************************************************************
*
* NOTE: SAS Enterprise Miner Licence required for
*     PROC DMDB, PROC DMREG, PROC NEURAL
* SAS MACROS ARE GIVEN BELOW TO PERFORM THE FOLLOWING FUNCTIONS:
* 1 Partition the data set into training, validation and test subsets
*     using sampling stratified by target(macro PARTITION).
* 2 Sample the training and validation subsets to create the required
*           N0:N1 ratio in the training and validation set
*           (macro RESAMPLE).
* 3 Impute missing values for inputs using mean on training data for
*           numeric inputs and most frequent class for categorical
*           inputs.
*           Makes use of utility macros
*           - count_v to count the number of inputs of type
*                 character or numeric
*           - fillit to write out the imputation code for numeric
*                 inputs
*                 to the SAS data step
*           - sqlit to put the mean of each numeric input into a
*                 separate macro variable
*           - fillchar to write out the imputation code for character
*                 inputs to the SAS data step
*           - frqchar to put the most frequent value of each
character
*                 input into separate macro variables
* 4 Create a data mining database (macro DMDB).
* 5 Fit logistic regression and score test set (macro REG).
* 6 Fit neural network and score test set (macro NEURAL).
* 7 Score test set with neural network early stopping weights
*     (macro NN_ES).
* 8 Calculate the evaluation criteria (macro EVALUATE).
* 9 Perform the (inner) loop for N0:N1 ratio for given run
*     (macro RATIO_LOOP).
*10 Perform the (outer) loop for run (macro ITER_LOOP).
*
* ADDITIONAL CODE AND MACROS:
* 1 PROC IML code to create bivariate normal sample
*           (equal variances case shown).
*           - SAS/IML licence required.
* 2 macro EVALUATE2 used instead of EVALUATE for artificial data sets
*     - criteria based on actual probability of class 1 calculated
*           in addition to other measures
*
* AUTHOR:         STEVEN HIRSCHOWITZ
* LAST UPDATE:    JULY 2008
*
*********************************************************************

/* PARTITION DATA  INTO TRAINING, VALIDATION AND TEST SUBSETS
      using sampling stratified by target: */
%macro partition(inds=kdd98.kdd98use,
id=controln,seed1=123,seed2=456,
            samprate1=0.4,samprate2=0.5);
proc surveyselect data=&inds seed=&seed1 out=_train
                  (drop=SelectionProb SamplingWeight)
                  samprate=(&samprate1,&samprate1);
      strata &target;
run;
proc sort data=_train; by &id; run;
data _tmp;
      merge _train(in=_a) &inds;
      by &id;
      if not _a;
run;
proc sort data=_tmp;
      by &target;
run;
proc surveyselect data=_tmp seed=&seed2 out=_valid
            (drop=SelectionProb SamplingWeight)
            samprate=(&samprate2,&samprate2);
      strata &target;
run;
proc sort data=_valid; by &id; run;
proc sql;create index &id on _tmp;  quit;
data test;
      merge _valid(in=_a) _tmp;
      by &id;
      if not _a;
run;
%mend;

/* COUNT THE NUMBER OF INPUTS FROM A GIVEN LIST
     VTYPE = iv for interval inputs or cv for catgorical inputs
*/
%macro count_v(vtype=iv);
%global n_&vtype;
%let n_&vtype=1;
%do %while (%scan(&&&vtype,&&n_&vtype) > "");
      %put &&n_&vtype %scan(&&&vtype,&&n_&vtype);
      %let n_&vtype = %sysevalf(&&n_&vtype+1);
%end;
%let n_&vtype = %sysevalf(&&n_&vtype-1);
%mend;

/* WRITE OUT THE MISSING VALUE STATEMENTS FOR INTERVAL INPUTS:*/
%macro fillit;
%do i = 1 %to %sysevalf(&n_iv);
      if missing(&&v&i) then &&v&i = &&m&i;
%end;
%mend;

/* PUT MEANS OF EACH INTERVAL VARIABLE INTO SEPARATE MACRO VARIABLE:
CALLS FILLIT MACRO TO REPLACE INTERVAL MISSING VALUES*/
%macro sqlit(train=train, valid=valid, test=test);
%do i = 1 %to %sysevalf(&n_iv);
      %global v&i;
      %let v&i = %scan(&iv,%sysevalf(&i));
      %global m&i;
      proc sql; select mean(&&v&i) into: m&i from &train; quit;
%end;
data &train;
      set &train;
      %fillit;
run;
data &valid;
      set &valid;
      %fillit;
```

```
run;
data &test;
      set &test;
      %fillit;
run;
%mend;

/* WRITE OUT THE MISSING VALUE STATEMENTS FOR CATEGORICAL INPUTS:*/
%macro fillchar;
%do i = 1 %to %sysevalf(&n_cv);
      if missing(&&c&i) then &&c&i = "&&cm&i";
%end;
%mend;

/* PUT MOST FREQUENT VALUE OF EACH CATEGORICAL VARIABLE INTO SEPARATE
      MACRO VARIABLE:
      CALLS FILLCHAR MACRO TO REPLACE MISSING VALUES*/
%macro frqchar(train=train, valid=valid, test=test);
%do i = 1 %to %sysevalf(&n_cv);
      %global c&i;
      %let c&i = %scan(&cv,%sysevalf(&i));
      %global cm&i;
      proc freq data=&train /*noprint*/ order=freq;
            tables &&c&i /out=&&c&i;
            where not missing(&&c&i);
      run;
      data &&c&i;
            set &&c&i(obs=1);
            call symput("cm&i",trim(left(&&c&i)));
      run;
%end;
data &train;
      set &train;
      %fillchar;
run;
data &valid;
      set &valid;
      %fillchar;
run;
data &test;
      set &test;
      %fillchar;
run;
%mend;

/* Sample the training and validation subsets to create
            the required N0:N1 ratio in the training and validation
set*/
%macro resample(train=&train,valid=&valid,ratio=1,prior=0.05,
                                    seed_t = 123, seed_v=456,classvar=NO);
      %put &ratio &prior;
      %local rate;
      %let rate=&ratio*&prior/(1-&prior);
      %if %sysevalf(&rate)>1 %then %let rate=1;
      %put &rate;
      proc sort data=&train;by &target; run;
      proc surveyselect data=&train out=train
rate=(%sysevalf(&rate),1)
                  seed=&seed_t;
            strata &target;
      run;
      %global nplus;
      proc sql; select count(*) into: nplus from train where
&target=1;
      proc sort data=&valid;by &target; run;
      proc surveyselect data=&valid out=valid
rate=(%sysevalf(&rate),1)
            seed=&seed_v;
            strata &target;
      run;
      %sqlit(train=train, valid=valid, test=test);
      %if "&classvar"="YES" %then %frqchar(train=train, valid=valid,
            test=test);
%mend;

/* CREATE DATA MINING DATABASE: */
%macro dmdb(train=train,dmdbcat=kdd98.dmdb_&ratio&run,classvar=NO);
proc dmdb batch data=&train
      dmdbcat=&dmdbcat;
      var &iv;
      class &target(desc) %if "&classvar"="YES" %then &cv; ;
      ;
      target &target;
run;
%mend;

/* FIT LOGISTIC REGRESSION, SCORE TEST DATA SET, STORE PARAMETER
      ESTIMATES AND FIT STATISTICS:*/
%macro reg(train=train, valid=valid, test=test, maxiter=100,ratio=1,
                        run=1,error=MBERNOULLI
,dmdbcat=kdd98.dmdb_&ratio&run,
                        forward=YES,classvar=NO);
proc dmreg data=&train
      dmdbcat=&dmdbcat estiter=1 outest=reg_outest
      validata=&valid;
      title "Ratio &ratio Run &run: Forward Logistic Regression";
      class &target %if "&classvar"="YES" %then &cv; ;
      nloptions maxiter=100 psummary maxtime=900;
      model &target = &iv %if "&classvar"="YES" %then &cv;
                              / error=&error
                              %if "&forward"="YES" %then selection =
forward; ;
      score data=&train out=reg_trainout outfit=reg_fit;
      score data=&valid out=nn_validout;
      score data=&test out=reg_testout;
      run;
quit;
data reg_outest;
      set reg_outest;
      ratio=&ratio;
      run=&run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;
proc append base=&lib..reg_outest data=reg_outest;
run;
data reg_fit;
      set reg_fit;
      ratio=&ratio;
      run=&run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;
proc append base=&lib..reg_fit data=reg_fit;
run;
%mend;

/* FIT NEURAL NETWORK, SCORE TEST DATA SET, STORE WEIGHTS AT EACH
            ITERATION AND FIT STATISTICS::*/
%macro neural(train=train, valid=valid, test=test,maxiter=100, hu1=3,
                        prelim=5, preiter=20, technique=levmar,
random=789,
                        ratio=1,run=1,maxtime=900,error=BER,

      dmdbcat=kdd98.dmdb_&ratio&run,classvar=NO,decay=0);
proc neural data=&train
      dmdbcat=&dmdbcat
      random=&random validata=&valid;
      title "Ratio &ratio Run &run: MLP &technique hidden=&hu1
            prelim=&prelim";
      netoptions decay=&decay;
      nloptions maxiter=&maxiter psummary maxtime=&maxtime;
      input &iv / level=interval id=i std=std;
      target &target / id=o level=nominal error=&error act=softmax
                        combine=linear;
      hidden &hu1 / id=h1 act=tanh combine=linear;
      connect i h1;
      %if "&classvar"="YES" %then input &cv / level=nominal id=n; ;
      %if "&classvar"="YES" %then connect n h1; ;
      connect h1 o;
      %if &prelim > 0 %then %str(prelim &prelim preiter=&preiter
            pretech=&technique outest=nn_prelim_outest;);
      train estiter=1 outest=nn_outest technique=&technique ;
      score data=&train out=nn_trainout outfit=nn_fit;
      score data=&valid out=nn_validout;
      score data=&test out=nn_testout;
      run;
quit;
data nn_outest;
      set nn_outest;
      ratio=&ratio;
      run=&run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;
proc append base=&lib..nn_outest_hu1_&hu1._hu2_&hu2 data=nn_outest;
run;
data nn_fit;
      set nn_fit;
      ratio=&ratio;
      run=&run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;
proc append base=&lib..nn_fit_hu1_&hu1._hu2_&hu2 data=nn_fit;
run;
%if &prelim > 0 %then %str(
data nn_prelim_outest;
      set nn_prelim_outest;
      ratio=&ratio;
      run=&run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;
proc append base=&lib..nn_prelim_outest_hu1_&hu1._hu2_&hu2
            data=nn_prelim_outest;
run;
);
%*put the early stopping weights in data set formin2:;
data formin1;
      set %if &prelim > 0 %then %str(nn_prelim_outest); nn_outest;
run;
proc sort data=formin1; by _VOBJ_; run;
data formin2;
      set formin1;
      if _n_ = 1;
run;
%mend;

/* SCORE TEST DATA SET USING EARLY STOPPING WEIGHTS:*/
%macro nn_es(train=train, valid=valid, test=test,maxiter=100, hu1=3,
                  prelim=5, preiter=20, technique=levmar, random=789,
                  ratio=1,run=1,maxtime=900,error=BER,
                  dmdbcat=kdd98.dmdb_&ratio&run,classvar=NO,decay=0);
proc neural data=&train
      dmdbcat=&dmdbcat
      random=&random validata=&valid;
      title "Ratio &ratio Run &run: MLP &technique hidden=&hu1
                  prelim=&prelim";
      nloptions maxiter=&maxiter psummary maxtime=&maxtime;
      input &iv / level=interval id=i std=std;
      target &target / id=o level=nominal error=&error act=softmax
                  combine=linear;
      hidden &hu1 / id=h1 act=tanh combine=linear;
      connect i h1;
      %if "&classvar"="YES" %then input &cv / level=nominal id=n; ;
      %if "&classvar"="YES" %then connect n h1; ;
      connect h1 o;
      initial inest=formin2;
      train technique=none;
      score data=&train out=nn_es_trainout outfit=nn_es_fit;
      score data=&test out=nn_es_testout;
      run;
quit;
data nn_es_fit;
      set nn_es_fit;
      ratio=&ratio;
      run=&run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;
proc append base=&lib..nn_es_fit_hu1_&hu1._hu2_&hu2 data=nn_es_fit;
run;
%mend;

/* CALCULATE EVALUATION CRITERIA ON THE TEST SET */
%macro
evaluate(method=nn,methodnum=1,rank=score_rank,out_dec=out_dec,
                              out_actval=out_actval,
                              out_auc=out_auc, out_overall =
out_overall,
                              out_mse = out_mse
                              );
            %local final_dec;
            %local final_auc;
            %local final_actval;
            %let final_dec=&lib..final_dec;
            %let final_auc=&lib..final_auc;
            %let final_actval=&lib..final_actval;
            %let final_mse=&lib..final_mse;
            %local testout;
            %let testout=&method._testout/*_&ratio&run*/ ;
            %local final_mpp;
            %let final_mpp=&lib..final_mpp;

data &method._testout;
      length method $7;
      set &method._testout(drop=p_&target.0 rename=(p_&target.1=P));
      method="&method";
      methodnum=&methodnum;

      %* adjust predicted probabilities for undersampling:;
    if &ratio > (1-&prior)/&prior then p_&target.1 = P;
       else p_&target.1
        = (P*&prior/&smplprop)/((P*&prior/&smplprop)
        + ((1-P)*(1-&prior)/(1-&smplprop)));
       p_&target.0 = 1-p_&target.1;

       %*squared error;
       if &target=0 then r2=p_&target.1**2; else r2=(1-
p_&target.1)**2;
       %*deviance residual;
       if &target=0 then do;
                  if p_&target.0 = 0 then devr2=-2*log(1E-20);
                        else devr2=-2*log(p_&target.0);
                  devr=-sqrt(devr2);
       end;
       else do;
                  if p_&target.1 = 0 then devr2=-2*log(1E-20);
                        else devr2=-2*log(p_&target.1);
                  devr=sqrt(devr2);
       end;
       %*pearson residual:;
       if p_&target.0 = 0 or p_&target.1 = 0 then
                  p_res = (&target-p_&target.1)/sqrt((1-1E-20)*1E-
20);
                  else p_res = (&target-p_&target.1)
                        /sqrt(p_&target.1*p_&target.0); *pearson
resid;

       ratio=&ratio;
       run=&run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;
%*store distribution of predicted probabilities:;
proc means data=&method._testout;
      var p_&target.1;
      output out=out_mean_pred_probs(compress=NO drop=_type_ _freq_)
                  min=min p1=p1 p5=p5 p10=p10 p25=p25
                  p50=p50 p75=p75 p90=p90 p95=p95 p99=p99 max=max
mean=mean;
run;
data out_mean_pred_probs;
      length method $7;
      set out_mean_pred_probs;
      method="&method";
      ratio=&ratio;
      run = &run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;
proc append base=&final_mpp data=out_mean_pred_probs force;
run;

%*compute and store AUC:;
proc rank data=&testout out=&testout._2 ;
      var P_&target.1;
      ranks &rank;
run;
proc sql;
      select sum(&rank) into: sum_rank_0 from &testout._2
            where &target=0;
      select sum(&rank) into: sum_rank_1 from &testout._2
            where &target=1;
quit;
data &out_auc;
      length method $7;
      method="&method";
      avg_rank_0 = &sum_rank_0 / &n0_tst;
      avg_rank_1 = &sum_rank_1 / &n1_tst;
      AUC = (&sum_rank_1 - &n1_tst*(&n1_tst +
1)/2)/(&n0_tst*&n1_tst);
      ratio=&ratio;
      run = &run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;

%*compute and store MSE (mean_r2) and deviance:;
proc means data=&testout noprint;
      var devr2 devr r2 ;
      output out=&out_mse N(devr2)=N
                  mean(devr2)=mean_devr2 sum(devr2) = deviance
                  mean(devr)=mean_devr mean(r2)=mean_r2 ;
run;
data &out_mse;
      length method $7;
      set &out_mse(drop=_type_ _freq_);
      method="&method";
      ratio=&ratio;
      run = &run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;
```

```
%*compute and store MSE (mean_r2) and deviance by actual target
class:;
proc means data=&testout noprint;
      var devr2 r2;
      class &target;
      output out=&out_mse.2 (where = (_type_=1))
                  mean(devr2)=mean_devr2 sum(devr2) = deviance
                  mean(r2)=mean_r2;
run;
data &out_mse.2;
      length method $7;
      set &out_mse.2(drop=_type_ _freq_);
      method="&method";
      ratio=&ratio;
      run = &run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;

%*compute and store Prop 1s per decile of predicted probability:;
proc rank data=&testout out=&testout._2 groups=10;
      var P_&target.1;
      ranks &rank;
run;
proc means data=&testout._2 noprint;
      var &target;
      class &rank;
      output out=means_&out_dec(where = (_type_=1)) mean=mean;
run;
proc transpose data=means_&out_dec(drop=_type_ _freq_)
            out=means_t_&out_dec prefix=prop_1_dec_;
      var mean;
      id &rank;
      where &rank ne .;
run;
proc means data=&testout._2 noprint;
      var &target;
      class &rank;
      output out=N_&out_dec(where = (_type_=1)) N=N;
run;
proc transpose data=N_&out_dec(drop=_type_ _freq_)
            out=N_t_&out_dec prefix=N_dec_;
      var N;
      id &rank;
      where &rank ne .;
run;
data t_&out_dec;
      length method $7;
      merge means_t_&out_dec N_t_&out_dec;
      method="&method";
      ratio=&ratio;
      run = &run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;

%*compute and store mean predicted probability by actual target
class:;
proc means data=&testout._2 noprint;
```

```
      var p_&target.1;
      class &target;
      output out=&out_actval(where = (_type_=1)) mean=mean;
run;
proc transpose data=&out_actval(drop=_type_ _freq_ )
      out=t_&out_actval prefix=mean_pred_actual_;
      var mean;
      id &target;
      where &target ne .;
run;
%*compute and store mean predicted probability overall:;
proc means data=&testout._2 noprint;
      var p_&target.1;
      output out=&out_overall(compress=NO drop=_type_ _freq_)
      mean=mean_predval
      ;
run;
data t_&out_actval;
      length method $7;
      merge t_&out_actval &out_overall;
      method="&method";
      ratio=&ratio;
      run = &run;
      nplus=&nplus;
      prior=&prior;
      hu1=&hu1;
      hu2=&hu2;
run;
proc append base=&final_dec data=t_&out_dec force;
run;
proc append base=&final_actval data=t_&out_actval force;
run;
proc append base=&final_auc data=&out_auc force;
run;
proc append base=&final_mse data=&out_mse force;
run;
proc append base=&final_mse._by_target data=&out_mse.2 force;
run;
%mend;

/* Perform the (inner) loop for N0:N1 ratio for given run */
%macro ratio_loop(run=1, nratios=1,startratio=1,classvar=NO);
%do tstratio = &startratio %to &nratios;
      %let ratio=%eval(2**(&tstratio-1));
      %let smplprop = (1/%sysevalf(&ratio+1));
      %put &ratio &smplprop &prior;
      %resample(train=_train,valid=_valid,ratio=&ratio,prior=&prior,
                              seed_t=5&run,seed_v=6&run
                              ,classvar=&classvar);
      %dmdb(train=train,dmdbcat=&lib..dmdb_&ratio&run,
                  classvar=&classvar);

      %neural(train=train, valid=valid, test=test,
                        maxiter=3000, hu1=&hu1, prelim=5, preiter=3,
                        technique=congra, random=7&run,
                        ratio=&ratio,run=&run,maxtime=2700,error=BER,
                        dmdbcat=&lib..dmdb_&ratio&run
                        ,classvar=&classvar,decay=0 );
      %let method=NN;
      %evaluate(rank=score_rank,out_dec=out_dec,out_actval=out_actval
,
                        out_auc=out_auc, out_overall = out_overall,
                        method=NN,methodnum=1)
      ;
      %let method=NN_ES;
      %nn_es(train=train, valid=valid, test=test,
                        maxiter=3000, hu1=&hu1, prelim=5, preiter=3,
                        technique=congra, random=7&run,
                        ratio=&ratio,run=&run,maxtime=2700,error=BER,
                        dmdbcat=&lib..dmdb_&ratio&run
                        ,classvar=&classvar,decay=0 );
      %evaluate(rank=score_rank,out_dec=out_dec,out_actval=out_actval
,
                        out_auc=out_auc, out_overall = out_overall,
                        method=NN_ES,methodnum=2)
      ;
      %reg(train=train, valid=valid, test=test,
maxiter=100,ratio=&ratio,
                  run=&run,error=MBERNOULLI
                  ,dmdbcat=&lib..dmdb_&ratio&run,forward=YES,
                  classvar=&classvar);
      %let method=Reg;
      %evaluate(rank=score_rank,out_dec=out_dec,out_actval=out_actval
,
                        out_auc=out_auc, out_overall = out_overall,
                        out_mse = out_mse,
                        method=Reg,methodnum=3)
      ;
%end;
%mend;

/* Perform the (outer) run loop*/
%macro iter_loop(firstiter=1,
lastiter=1,nratios=&nratios,startratio=1,

      classvar=NO,samprate1=0.4,samprate2=0.5);
proc sort data=&inds force; by &target; run;
proc sql;create index &id on &inds; quit;
%do run = &firstiter %to &lastiter;
      proc printto log=mylog print=myout;
      run;
      %partition(inds=&inds, id=&id,seed1=123&run,seed2=345&run,
                        samprate1=&samprate1,samprate2=&samprate2);
      proc sql;
      select count(*) into: n0_tst from test where &target=0;
      select count(*) into: n1_tst from test where &target=1;
      quit;
      %ratio_loop(run=&run,nratios=&nratios,startratio=&startratio,
                              classvar=&classvar);
      proc printto;
      run;
      %put FINISHED Run &run;
%end;
%put FINISHED!!;
%put YAY!!;
%mend;

/* EXAMPLE OF CALLING THE MACROS */
%let iv=
EXPTD_CR_TRNVR_A AVE_VAL_AUTO_CR_L3M    TOT_NET_TURNOVER
AVG_NET_TRN_L3M ATM_DPOST_6M_Q MAX_BALANCE MINIMUM_BAL_L6M
TOT_DEPOSIT_SAVGS_BAL ADRB_6M_A ABB_6M_A_M CAR_SO_CONF_LIMIT
```

```
NET_LENDING_LIMIT
TOTAL_UNSECURED_LIMIT MNTH_SIN_LAST_LIM_CHN TOT_SECURED_LENDING NUM_I
NUM_L NUM_M NUM_S NUM_T TOTACCS DAYS_EXCESS NO_MTHS_IN_EXCESS_L6M
AVE_REG_DR_COMMIT_L6M  AVG_DBT_TRN_L3M NUMBER_DEBIT_TRANSACTIONS
ACNT_AGE
TIME_WITH_BANK CUST_AGE  MNTH_SIN_LST_CRED_OPENED  FEE_INCOME
SSWCH_FEE_6M_A DISPOSABLE_INCOME
;
%count_v(vtype=iv);
%let cv=
ACCOMMODATION_TYPE CUST_C GENDER MARITAL_STATUS SAL_TYPE
;
%count_v(vtype=cv);

%let target=new_H;
%let id=new_cust_n;
%let inds=mdl0706.use; *input data set;
*calculate prior:;
proc sql; select sum(&target=1)/count(*) into: prior from &inds;
quit;
%let hu1=5; *num hidden units hidden layer 1;
%let hu2=0; *num hidden units hidden layer 2 - 0 if no hidden layer
2;
%let lib=mdl07064;
%let dir =C:\steve\masters\Oversampling\sasdata\mdl0706v4;
dm 'log;clear;output;clear;';
title;
filename mylog "&dir\log_output\log_1_hu1_&hu1._hu2_&hu2..log";
filename myout "&dir\log_output\out_1_hu1_&hu1._hu2_&hu2..out";
proc printto log=mylog print=myout new;
run;
%iter_loop(firstiter=1, lastiter=30
,startratio=1,nratios=7,classvar=YES,
                  samprate1=0.4,samprate2=0.5);
proc printto;run;
%put FINISHED!!;
%put YAY!!;

* samprate1 and samprate2 shown create a 40:30:30
            training:validation:test partition;


/*********************************************************************
**
* ADDITIONAL CODE AND MACROS:
* 1 PROC IML code to create bivariate normal sample
*           (equal variances case shown).
*           - SAS/IML licence required.
* 2 macro EVALUATE2 used instead of EVALUATE for aftificial data sets
*      - criteria based on actual probability of class 1 calculated
*           in addition to other measures
*********************************************************************
*/

/* Create bivariate normal sample */
%let Nplus=1000;
%let Nminus=19000;
proc iml;
N=&Nplus;
Mean = {15 20};
Corr = {1 0.3,
                  0.3 1};
Var = {4 4};
%* create the covariance matrix;
Cov = Corr # sqrt(Var` * Var);
  call vnormal(save, Mean, Cov, N, 123);

  col={ "x1" "x2"};
  create plus from save [colname=col];
  append from save;
quit;

proc iml;
N=&Nminus;
Mean = {13 18};
Corr = {1 0.3,
                  0.3 1};
Var = {4 4};
/*create the covariance matrix*/
Cov = Corr # sqrt(Var` * Var);
  call vnormal(save, Mean, Cov, N, 456);

  col={ "x1" "x2"};
  create minus from save [colname=col];
  append from save;
quit;

%let target=target;
data plus_minus;
      retain id;
      set plus(In=_a) minus(In=_b);
      if _a then &target=1; else &target=0;
      id+1;
run;

/* MACRO EVALUATE2 - CALCULATES EVALUATION CRITERIA FOR ARTIFICIAL
DATA */

%macro
evaluate2(method=nn,methodnum=1,rank=score_rank,out_dec=out_dec,
                              out_actval=out_actval,
                              out_auc=out_auc, out_overall =
out_overall,
                              out_mse = out_mse
                              );
            %local final_dec;
            %local final_auc;
            %local final_actval;
            %let final_dec=&lib..final_dec;
            %let final_auc=&lib..final_auc;
            %let final_actval=&lib..final_actval;
            %let final_mse=&lib..final_mse;
            %local testout;
            %let testout=&method._testout/*_&ratio&run*/ ;
            %local final_mpp;
            %let final_mpp=&lib..final_mpp;

data &method._testout(drop=pi s1 s2 s11 s22 z r rsq sqrt u1 u2);
      length method $7;
      set &method._testout(drop=p_&target.0 rename=(p_&target.1=P));
      method="&method";
      methodnum=&methodnum;
```

```
       %* adjust predicted probabilities for undersampling:;
     if &ratio > (1-&prior)/&prior then p_&target.1 = P;
        else p_&target.1
         = (P*&prior/&smplprop)/((P*&prior/&smplprop)
         + ((1-P)*(1-&prior)/(1-&smplprop)));
        p_&target.0 = 1-p_&target.1;

        %* calculate the actual probability of rare class:;
             pi = 3.141592654;

             r = 0.3;
             rsq = r**2;
             sqrt = sqrt(1-rsq);
             u1 = 15; u2 = 20; s11 = &s11_p; s22 = &s22_p;
             s1 = sqrt(s11); s2=sqrt(s22);
             z=(x1-u1)**2/s11 - 2*r*(x1-u1)*(x2-u2)/(s1*s2)+(x2-
u2)**2/s22;
             f1_x = 1/(2*pi*s1*s2*sqrt)*exp(-z/(2*(1-rsq)));

             u1 = 13; u2 = 18; s11 = &s11_n; s22 = &s22_n;
             s1 = sqrt(s11); s2=sqrt(s22);
             z=(x1-u1)**2/s11 - 2*r*(x1-u1)*(x2-u2)/(s1*s2)+(x2-
u2)**2/s22;
             f0_x = 1/(2*pi*s1*s2*sqrt)*exp(-z/(2*(1-rsq)));

             p_act = f1_x*&prior / (f1_x*&prior + f0_x*(1-&prior));

        %* calculate squared difference between actual and predicted
                     probability:;
             err_100 = 100*(p_&target.1 - p_act);
             abs_err_100 = abs(err_100);
             sq_err_10K = err_100**2;

        %* for baseline for squared difference:;
             err_random_100 = 100*(&prior - p_act);
             abs_err_random_100 = abs(err_random_100);
             sq_err_random_10K = err_random_100**2;

        %*squared error;
        if &target=0 then r2=p_&target.1**2; else r2=(1-
p_&target.1)**2;

        %*deviance residual;
        if &target=0 then do;
                   if p_&target.0 = 0 then devr2=-2*log(1E-20);
                         else devr2=-2*log(p_&target.0);
                   devr=-sqrt(devr2);
        end;
        else do;
                   if p_&target.1 = 0 then devr2=-2*log(1E-20);
                         else devr2=-2*log(p_&target.1);
                   devr=sqrt(devr2);
        end;

        %*pearson residual:;
        if p_&target.0 = 0 or p_&target.1 = 0 then
                         p_res = (&target-p_&target.1)/sqrt((1-1E-
20)*1E-20);
                   else p_res = (&target-p_&target.1)
```

```
                        /sqrt(p_&target.1*p_&target.0); *pearson
resid;

       ratio=&ratio;
       run=&run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;

%*store distribution of predicted probabilities:;
proc means data=&method._testout;
      var p_&target.1;
      output out=out_mean_pred_probs(compress=NO drop=_type_ _freq_)
                  min=min p1=p1 p5=p5 p10=p10 p25=p25
                  p50=p50 p75=p75 p90=p90 p95=p95 p99=p99 max=max
mean=mean;
run;
data out_mean_pred_probs;
      length method $7;
      set out_mean_pred_probs;
      method="&method";
      ratio=&ratio;
      run = &run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;
proc append base=&final_mpp data=out_mean_pred_probs force;
run;

%*compute and store Mean Squared Difference between Actual and
predicted probability:;
proc means data=&method._testout;
      var sq_err_10K;
      output out=out_mean_sq_err_10K(compress=NO drop=_type_ _freq_)
                  min=min p1=p1 p5=p5 p10=p10 p25=p25
                  p50=p50 p75=p75 p90=p90 p95=p95 p99=p99 max=max
mean=mean;
run;
data out_mean_sq_err_10K;
      length method $7;
      set out_mean_sq_err_10K;
      method="&method";
      ratio=&ratio;
      run = &run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;
proc append base=&lib..final_mse_10K data=out_mean_sq_err_10K force;
run;

%*get baseline for MSD:;
proc means data=&method._testout;
      var sq_err_random_10K;
      output out=out_mean_sq_err_random_10K(compress=NO
            drop=_type_ _freq_)
                  min=min p1=p1 p5=p5 p10=p10 p25=p25
                  p50=p50 p75=p75 p90=p90 p95=p95 p99=p99 max=max
mean=mean;
```

```
run;
data out_mean_sq_err_random_10K;
      length method $7;
      set out_mean_sq_err_random_10K;
      method="&method";
      ratio=&ratio;
      run = &run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;
proc append base=&lib..final_mse_random_10K
            data=out_mean_sq_err_random_10K force;
run;

*compute and store AUC:;
proc rank data=&testout out=&testout._2 ;
      var P_&target.1;
      ranks &rank;
run;
proc sql;
      select sum(&rank) into: sum_rank_0 from &testout._2
            where &target=0;
      select sum(&rank) into: sum_rank_1 from &testout._2
            where &target=1;
quit;
data &out_auc;
      length method $7;
      method="&method";
      avg_rank_0 = &sum_rank_0 / &n0_tst;
      avg_rank_1 = &sum_rank_1 / &n1_tst;
      AUC = (&sum_rank_1 - &n1_tst*(&n1_tst +
1)/2)/(&n0_tst*&n1_tst);
      ratio=&ratio;
      run = &run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;

%*compute and store MSE (mean_r2) and deviance:;
proc means data=&testout noprint;
      var devr2 devr r2 ;
      output out=&out_mse N(devr2)=N
                  mean(devr2)=mean_devr2 sum(devr2) = deviance
                  mean(devr)=mean_devr mean(r2)=mean_r2 ;
run;
data &out_mse;
      length method $7;
      set &out_mse(drop=_type_ _freq_);
      method="&method";
      ratio=&ratio;
      run = &run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;
%*compute and store MSE (mean_r2) and deviance by actual target
class:;
proc means data=&testout noprint;
      var devr2 devr r2 ;
      class &target;
      output out=&out_mse.2 (where = (_type_=1))
                  mean(devr2)=mean_devr2 sum(devr2) = deviance
                  mean(devr)=mean_devr mean(r2)=mean_r2;
run;
data &out_mse.2;
      length method $7;
      set &out_mse.2(drop=_type_ _freq_);
      method="&method";
      ratio=&ratio;
      run = &run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;

%*compute and store Prop 1s per decile of predicted probability:;
proc rank data=&testout out=&testout._2 groups=10;
      var P_&target.1;
      ranks &rank;
run;
proc means data=&testout._2 noprint;
      var &target;
      class &rank;
      output out=means_&out_dec(where = (_type_=1)) mean=mean;
run;
proc transpose data=means_&out_dec(drop=_type_ _freq_)
            out=means_t_&out_dec prefix=prop_1_dec_;
      var mean;
      id &rank;
      where &rank ne .;
run;
proc means data=&testout._2 noprint;
      var &target;
      class &rank;
      output out=N_&out_dec(where = (_type_=1)) N=N;
run;
proc transpose data=N_&out_dec(drop=_type_ _freq_)
            out=N_t_&out_dec prefix=N_dec_;
      var N;
      id &rank;
      where &rank ne .;
run;
data t_&out_dec;
      length method $7;
      merge means_t_&out_dec N_t_&out_dec;
      method="&method";
      ratio=&ratio;
      run = &run;
      nplus=&nplus;
      hu1=&hu1;
      hu2=&hu2;
run;

%*compute and store mean predicted probability by actual target
class:;
proc means data=&testout._2 noprint;
      var p_&target.1;
      class &target;
      output out=&out_actval(where = (_type_=1)) mean=mean;
run;
```

```
proc transpose data=&out_actval(drop=_type_ _freq_ )
            out=t_&out_actval prefix=mean_pred_actual_;
      var mean;
      id &target;
      where &target ne .;
run;
%*compute and store mean predicted probability overall:;
proc means data=&testout._2 noprint;
      var p_&target.1;
      output out=&out_overall(compress=NO drop=_type_ _freq_)
            mean=mean_predval
      ;
run;
data t_&out_actval;
      length method $7;
      merge t_&out_actval &out_overall;
      method="&method";
      ratio=&ratio;
      run = &run;
      nplus=&nplus;
      prior=&prior;
      hu1=&hu1;
      hu2=&hu2;
run;
proc append base=&final_dec data=t_&out_dec force;
run;
proc append base=&final_actval data=t_&out_actval force;
run;
proc append base=&final_auc data=&out_auc force;
run;
proc append base=&final_mse data=&out_mse force;
run;
proc append base=&final_mse._by_target data=&out_mse.2 force;
run;
%mend;
```