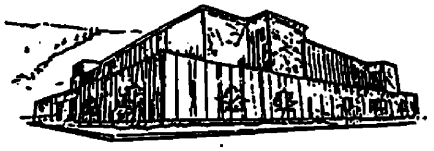
2003

Estimating classification accuracy using probability of correct classification estimates

Jiyan Du
The University of Montana



Maureen and Mike
MANSFIELD LIBRARY

The University of
Montana

Estimating Classification Accuracy
Using Probability of Correct Classification Estimates

By

Jiyan Du

B.A Beijing Polytechnic University, China 1998

Presented in partial fulfillment of the requirements

For the degree of

Master of Arts

The University of Montana

September 2003

Approved by:

Brian Stark

Chairman

Dean Graduate School

9-10-03

Date

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

Jiyan Du Master, September 2003 Mathematical Sciences

Estimating Classification Accuracy Using Probability of Correct Classification Estimates

Chair: Brian Steele BMS

The objective of classification problem is to determine a rule that will predict the group membership of an observation whose membership is unknown. An important second aspect of the classification problem is assessing the accuracy of classification rules. Here, we discuss several estimators of classifier accuracy.

These estimates include simple means of maximum posterior probability (MPP) estimators and calibrated versions thereof. Calibration functions are estimated by regressing cross validation (CV) outcomes on the MPP estimates. A simulation study was conducted to compare ordinary CV estimates and MPP-based estimates. In general, CV estimator and linear calibrated MPP estimator are better than the other two, and linear calibrated MPP estimator is best with respect to root mean square error (RMSE).

Contents

Acknowledgments

First, I want to say thanks to my parents. They have always supported me and shared in my happiness.

I want to thank Dr. Brian Steele, Dave Patterson and Roly Redmond. They were willing to be the committee members who helped me finish my paper. Dr. Steele gave me a great help in my studying as my advisor and I got many helpful comments and suggestions about my paper from him.

I also want to thank Dr. Jon Graham, Thomas Tonev, Rudy Gideon, William R. Derrick, Gregory St.George, Karel Stroethoff, etc. I took some classes from them and I am really appreciated for their help in my studying.

I want to thank Mark Heaphy, Kathy Gray, Neil Templeton, Joyce Schlieter, Jane Wilson, Yaling Hsu and Jeff Rosky . They gave me a lot of help as my classmates or when I work with them as a teaching assistant.

I want to thank Mr. and Mrs. Wilder. They are my good friends. They introduced me to different traditions and culture here and I even shared the harvest of vegetables from their backyard.

I want to thank Mr. Seva Kumar. He gave me a great help in my internship. Then I was able to enjoy my work and finish my paper at the same time.

Thanks for all of my friends. They make my life happier.

1.Introduction

"Depending on the problem, the basic purpose of a classification study can be either to produce an accurate classifier or to uncover the predictive structure of the problem. If we are aiming at the latter, then we are trying to get an understanding of what variables or interactions of variables drive the phenomenon-- that is, to give simple characterizations of the conditions that determine when an object is in one class rather than another. These two are not exclusive. Most often, in our experience, the goals will be both accurate prediction and understanding. Sometimes one or the other will have greater emphasis."(Breiman et al, 1984).

After we define a classification rule and use it to classify the observations, inevitably, some of the units will be incorrectly classified. So it is important to assess the accuracy. Accuracy assessment is the topic of this paper.

In the following discussion, I will introduce several estimators of accuracy based on cross-validation. They are ordinary cross-validation estimator, maximum posterior probability (MPP) estimator and linear and logistic calibration of MPP estimators.

A simulation study was conducted to compare such estimates. The key research question is, "Are the MPP-based estimates as good as the CV estimates?"

2. Classifiers

Suppose there is a population P partitioned as c classes, groups, or subpopulations $G_g, g = 1,...,c$. The objective is to determine a rule that will predict the group membership of an observation whose membership is unknown. First, we randomly select a sample $X = \{x_1, x_2..., x_n\}$ from P. The sample is often called a training sample in recognition that the rule will be trained on these data. In addition, one or more variables

(usually called covariates) are measured on every observation in the sample. These covariates are assumed to differ among groups, and hence, have some predictive value for classification. An important second aspect of the classification problem is assessing the accuracy of the rule; that is, it is important to obtain an estimate that the rule will correctly classify a population unit.(Steele, 2003, unpublished notes).

The *i*th observation in X is a pair denoted by $x_i = (t_i, y_i)$ where t_i is a covariate vector and y_i is a group label identifying group membership. Let $x_0 = (t_0, y_0)$ denote an unclassified observation belonging to *P*. The covariate vector t_0 is observed but the group label y_0 is unobserved. .(Steele, 2003, unpublished notes).

In some cases, a researcher may have prior knowledge as to how likely it is that a randomly selected observation would come from each of the two or more groups. The prior probability of membership in a group is the probability that a randomly selected observation will be a member of the group. After collecting the training sample and forming the rule, the probability of group membership varies with the predictor variables (unless these variables are useless as predictors). The conditional probabilities of group membership, given the predictor variables, are the posterior probabilities. The prior probability that x_0 belongs to G_g is denoted by $\pi_g = P(y_0 = g)$ whereas the posterior probability of membership in G_g is $P(y_0 = g \mid t_0)$.

A general approach to classification can be formulated by treating classifiers (classification rules) as estimators of the probabilities of group membership, i.e., of $P(y_0 = g \mid t_0), g = 1,...c.$ The assignment of group membership to x_0 is the same for all

classifiers: assign x_0 to the group with the largest posterior probability. Let η denote the classifier, the rule is

$$\eta(x_0) = \arg\max_g P(y_0 = g \mid t_0).$$

The formulation of this optimal rule is based on the criterion of minimizing the probability that an unclassified observation will be incorrectly classified by the rule, i.e., minimize $P(\eta(x_0) \neq y_0)$.(Steele, 2003, unpublished notes)

2.1 Bayes rule

Definition 2.1. η is a Bayes rule if for any other classifier η', $P(\eta(x) \neq y) \leq P(\eta'(x) \neq y)$. Then the Bayes misclassification rate is $R_B = P(\eta(x) \neq y)$. It minimizes the probability of misclassification.

Theorem 2.2 The Bayes rule η is defined as $\eta(x_0) = \arg\max_g P(y_0 = g \mid t_0)$ (2.1); In each group $j, j = 1,...,c$, there is density function $f_j(x), x \in G_j$, i.e.

$P(\eta(x) = j \mid y = j) = \int_{x|\eta(x)=j} f_j(x)dx$, the Bayes misclassification rate is

$R_B = 1 - \int[\max_j f_j(x)\pi_j]dx$ (2.2).

Proof:

$$P(\eta(x) = y) = \sum_{j=1}^{c} P(\eta(x) = j \mid y = j)P(y = j)$$

$$= \sum_{j=1}^{c} P(\eta(x) = j \mid y = j)\pi_j$$
$$= \sum_{j=1}^{c} \int_{x|\eta(x)=j} f_j(x)\pi_j dx$$
$$= \int[\sum_{j=1}^{c} \Psi(\eta(x) = j) f_j(x)\pi_j]dx$$

$\Psi(\eta(x) = j)$ is an indicator, $\Psi(\eta(x) = j) = \begin{cases} 1, & if \quad \eta(x) = j \\ 0, & if \quad \eta(x) \neq j \end{cases}$

For a fixed value of x

$\sum_{j=1}^{c} \Psi(\eta(x) = j) f_j(x)\pi_j \leq \max[f_j(x)\pi_j]$.

and equality is achieved if $\eta(x)$ equals that j for which $f_j(x)\pi_j$ is a maximum.

Therefore, the rule η given in (2.1) has the property that for any other classifier η',

$P(\eta'(x) = y) \leq P(\eta(x) = y) = \int \max_j [f_j(x)\pi_j] dx$.

This shows that η is a Bayes rule and establishes (2.2) as the correct equation for the Bayes misclassification rate.

Although η is called the Bayes rule, it is also recognizable as a maximum likelihood rule: Classify x as that j for which $f_j(x)\pi_j$ is maximum. Please note that (2.1) does not uniquely define η on points x such that $\max_j f_j(x)\pi_j$ is achieved by two or more different j's. In this situation, we can define η arbitrarily to be any one of the maximizing j's or use other methods to break the ties.

In practice, neither the π_j nor the $f_j(x)$ are known. The π_j can either be estimated as the proportion of class j in training sample or their values supplied through other knowledge about the problem. There are several ways to estimate $f_j(x)$.

2.2 Linear discriminant classifier

Discriminant analysis assumes that all $f_j(x)$ are multivariate normal densities with common covariance matrix Σ and different mean vectors $\{\mu_j\}$. Estimating Σ and

the $\{\mu_j\}$ in the usual way gives estimates $\hat{f}_j(x)$ of the $f_j(x)$. Randomly select a sample G as the training sample, in $G_j, X_j = \{x_1, x_2 ..., x_{n_j}\}$, the estimates are

$$\hat{\mu}_j = \frac{1}{n_j}\sum_{r=1}^{n_j} t_r, \quad \hat{\Sigma}_j = \frac{1}{n_j - 1}\sum_{r=1}^{n_j}(t_r - \hat{\mu}_j)(t_r - \hat{\mu}_j)'.$$

The pooled covariance matrix is $\hat{\Sigma} = \frac{\sum_{j=1}^{c}[(n_j - 1)\hat{\Sigma}_j]}{\sum_{j=1}^{c}(n_j - 1)}$. These are substituted into the Bayes optimal rule $\eta(x_0) = \arg\max_g \hat{P}(y_0 = g \mid t_0) = \arg\max_g \hat{f}_g(x_0)\pi_g$. (Breiman et al, 1984).

2.3 *k*-NN classifier

Nearest neighbor discriminant analysis is a nonparametric discriminant procedure. It is developed without any distribution assumption. It uses the distances between pairs of observation vectors.

For any new observation x_0, we can find the k nearest neighbors to x_0 in the training sample. Classify x_0 as class g if most of the neighbors are in group g.

The k-NN estimate of $P_j(x_0)$ is $P_j^{kNN}(x_0) = \frac{1}{k}\sum_{i=1}^{k}\Psi(y_i = j)$. $\Psi(E)$ is the indicator function of the event E. $\Psi(y_i = j) = \begin{cases} 1, & if \quad y_i = j \\ 0, & if \quad y_i \neq j \end{cases}$. If there is a tie, we can increase the neighborhood size, and recompute the estimate of P_j^{kNN} until the ties are broken (Steele Patterson and Redmond, 2003).

3.Estimators

After we construct a classification rule and use it to classify the units in a test sample, some of the units will be incorrectly classified inevitably. So it is important to assess the accuracy.

3.1 Max posterior probability estimator

Them3.1 The probability that the rule correctly classifies the observation is equal to the max probability of group membership.

Suppose x_0 is a randomly selected observation, the probability that the rule is correct is

$$P[(\eta(x_0) = y_0)] = P[\arg\max_g P(y_0 = g \mid t_0) = y_0]$$

$$= \sum_{j=1}^{c} \Psi[\eta(x_0) = j] P(y_0 = j \mid t_0)$$

We define that all of the indicator variables in this sum are 0 except the indicator of the group for which the probability of membership, $P(y_0 = g \mid t_0)$, is maximal. For that group, say group g^*, $P(y_0 = g^* \mid t_0) = \max_g P(y_0 = g \mid t_0)$. Hence,

$$P[(\eta(x_0) = y_0)] = \max_g P(y_0 = g \mid t_0).$$

Repeat it n times, we get $acc = \frac{1}{n} \sum_{i=1}^{n} P[(\eta(x_i) = y_i)]$.

The limitation with this formula is that $P_g(x_0) = P(y_0 = g \mid t_0), g = 1,...c$ is not known and must be estimated with negligible bias. One approach is to use the plug-in estimator, i.e., compute $\hat{P}[\eta(x_0) = y_0] = \max_g \hat{P}_g(x_0)$, where $\hat{P}_g(x_0) = \hat{P}(y_0 = g \mid t_0)$ is an estimate derived from the classifier. (Steele, 2003, unpublished notes).

3.2 Ordinary k-fold cross validation estimator

It can be described in the following manner:

Divide the data set into k subsets of as nearly equal size as possible. Remove the first subset from the data set, form a classification rule based on all of the remaining data, use this rule to classify the first subset. Next, replace the first subset and remove the second subset from the data set, form a classification rule based on all of the remaining data, use this rule to classify the second subset, and noting whether a particular observation in that subset would be correctly classified by a rule formed from all of the remaining data.

$$\text{Set } \Psi(y_i = \hat{y}_i) = \begin{cases} 1, & if \quad y_i = \hat{y}_i \\ 0, & if \quad y_i \neq \hat{y}_i \end{cases}, \quad \hat{p}^{CV} = \frac{\sum \Psi(y_i = \hat{y}_i)}{n}, \; n \text{ is the sample size.}$$

3.3 Calibrating the Estimated Probabilities of Correct Classification

It is important to recognize that there may be some bias when we use the plug in estimates instead of the true probabilities. This is the reason that we calibrate the probability estimates.

Calibration is carried out by regressing the binary leave-one-out outcomes $\Psi[\eta(x_i) = y_i], i = 1,...,n,$ on the leave-one-out probability estimates of correct classification $\hat{P}(\eta(x_i) = y_i)$ to obtain a calibration coefficient. Linear and logistic regression can be used to derive calibration functions from the training set. The calibration coefficient is used to calibrate the probability estimates according to the calibration function.

3.3.1 Linear calibration

To set up the calibration functions, let $\hat{p}_i = \hat{P}[\eta_i(x_i) = y_i]$ denote the estimated probability that $x_i \in X_n$ is correctly classified by the holdout η_i, and

$O_i = \Psi[\eta_i(x_i) = y_i], i = 1,...n,$ denote the outcome of classifying x_i by η_i. For $x_i \in X_n$, the linear calibration function specifies that the calibrated of $P[\eta_i(x_i) = y_i]$ is

$$\hat{p}_i^{lin} = \begin{cases} \hat{\beta}\hat{p}_i, & if \quad \hat{\beta}\hat{p}_i < 1 \\ 1, & if \quad \hat{\beta}\hat{p}_i, \geq 1 \end{cases}$$

The coefficient $\hat{\beta}$ is determined by minimizing $\sum(O_i - \beta\hat{p}_i)^2$ with respect to β;

$$\frac{\partial\sum(O_i - \beta\hat{p}_i)^2}{\partial\beta} \underline{\underline{set}} \ 0$$

$$\Rightarrow 2\sum(O_i - \beta\hat{p}_i)\ \hat{p}_i = 0$$

$$\Rightarrow \hat{\beta} = \sum O_i\hat{p}_i / \sum\hat{p}_i^2$$

(Insert figure 1 here.)

So $\hat{p}^{lin} = \frac{\sum\hat{p}_i^{lin}}{n}$, n is the sample size.

3.3.2 Logistic calibration

Logistic regression is justified under the assumption that $O_i, i = 1,...n,$ are independent Bernoulli random variables with expections $p_i = P[\eta_i(x_i) = y_i]$,i.e., $O_i \sim B(p_i)$.

Then, the logistic calibration model is

$$\log(\frac{P[\eta_i(x_i) = y_i \mid x_i]}{1 - P[\eta_i(x_i) = y_i \mid x_i]}) = \beta\log(\frac{\hat{p}_i}{1 - \hat{p}_i})$$

$$\Rightarrow \frac{p_i}{1 - p_i} = (\frac{\hat{p}_i}{1 - \hat{p}_i})^{\beta}$$

$$\Rightarrow \hat{p}_i^{\log} = \{1 + [(1 - \hat{p}_i)/\hat{p}_i]^{\hat{\beta}}\}^{-1}$$

where the calibration coefficient $\hat{\beta}$ is computed by logistic regression (Steele Patterson and Redmond, 2003).

So $\hat{p}^{\log} = \frac{\sum \hat{p}_i^{\log}}{n}$, n is the sample size.

4. Simulation design

The simulation design is the same as used by Steele and Patterson (2000). As they describe, "the simulated sets were randomly chosen, respectively, from the bivariate distributions $\frac{2}{5}N(\mu_{11},\sigma_1 I_2)+\frac{3}{5}N(\mu_{12},\sigma_1 I_2)$, $N(\mu_2,\sigma_2 I_2)$ and $N(\mu_3,\sigma_3 I_2)$ where $\mu_{11}^T=(3,3),\sigma_1=1.5,\mu_{12}^T=(7,7),\mu_2^T=(4,6),\sigma_2=2,\mu_3^T=(7.5,3.5),\sigma_3=3.$"

The sample size is 10,000 which is large enough that we can regard it as an infinite population. Sample of n ={100,200,300,400,600,1000} observations were drawn from this population of 10,000 at random. A classification rule was constructed from each training sample.

The classifier was used to estimate the posterior probabilities of group membership. An observation was assigned to the group that gave the largest posterior probability estimates.

Five-fold, ten-fold and n-fold cross validation were used in the calculation of the post probability estimates. Four accuracy estimates were computed:

Ordinary Cross-Validation estimate = $\frac{1}{n}\sum \Psi(\hat{y}_i = y_i)$

Max Posterior Probability estimate= $\frac{1}{n}\sum MPP_i$

Linear calibration estimate = $\hat{\beta}^{lin}\frac{1}{n}\sum MPP_i$

Logistic calibration estimate = $\frac{1}{n}\sum\{1+[(1-MPP_i)/MPP_i]^{\hat{\beta}^{\log}}\}^{-1}$

This procedure was repeated 1,000 times, and the averages were computed. The entire population of 10,000 observations was classified to get the true accuracy. Compare the true accuracy to the estimates, we use two measures of performance.

$Bias=\frac{\sum acc_k - a\hat{c}c_k}{1{,}000}$ and root mean square error $RMSE=\sqrt{\frac{\sum (acc_k - a\hat{c}c_k)^2}{1{,}000}}$.

Here acc_k is the true accuracy and $a\hat{c}c_k$ is the estimated accuracy for the kth repetition. The key research question is ," Are the MPP-based estimates as good as the CV estimates?"

5. Result and discussion

We use two classifiers to classify the observations, and use six sample sizes to make simulations. The following is the result about the changes of Bias and RMSE of these four estimators. We can find some tendency from Figure 3 to Figure 14. In each figure, there are four lines, which represent the changes for the estimates from each estimator with the increasing of sample sizes.

Figure 3,4,5 show the Bias changes for LDA classifer for 3 different Cross-Validation.

Figure 6,7,8 show the Rmse changes for LDA classifer for 3 different Cross-Validation.

Figure 9,10,11 show the Bias changes for kNN classifer for 3 different Cross-Validation.

Figure 12,13,14 show the RMSE changes for kNN classifer for 3 different Cross-Validation.

1. We get six true accuracy estimates for each classifier. With the increasing of the sample size, from 100 to 1000, the true accuracy estimate increases too. The values increase from 60.15 to 61.78 if we use LDA classifier and from 63.957 to 66.428 for k-NN classifier.

(Insert figure 2 here.)

From the plot, we can see *k*-NN classifier always gets a higher accuracy than LDA classifer.

2.Compare the bias.

First look at the results from LDA classifier. With the increase of sample size from 100 to 1000, the bias for CV and Linear calibration (LinC) estimates are very stable. MPP and Logistic calibration (LogC) estimates are not stable at all. Both of them increase with the increasing of the sample sizes.

Under the same sample size, the biases for CV and LinC are much smaller than the other two. And CV estimate is a little better than LinC estimate.

(Insert figure 3,4,5 here.)

We can get the similar result from *k*-NN classifier.

(Insert figure 6,7,8 here.)

3.Compare the RMSE .

First look at the results from LDA classifier. With the increasing of the sample sizes from 100 to 1000, the RMSE values for CV and LinC estimates decrease. The values for MPP and Logistic calibration (LogC) estimates are very stable. They don't decrease with the sample size.

Under the same sample size, the RMSE values for CV and LinC estimates are much smaller than the other two. And LinC gets smaller RMSE than CV estimator.

(Insert figure 9,10,11 here.)

We can get the similar result from *k*-NN classifier.

(Insert figure 12,13,14 here.)

4.In general, CV estimator and linear calibrated MPP estimator are better than the other two, and linear calibrated MPP estimator is best with respect to root mean square error.

6. References

Steele,B.M. Patterson, D.A. and Redmond, R.L Estimating Thematic Map Accuracy Without a Probability Test Sample

Steele,B.M. and Patterson, D.A. Ideal bootstrap estimation of expected prediction error for k-nearest neighbor classifiers: Applications for classification and error assessment

Johnson, D. E. (1998) Applied Multivariate Methods for Data Analysis

Breiman, L. Friedman, J.H. and Olshen, R.A. Stone, C.J.(1984) Classification and Regression Trees

Cacoullos, T. (1972) Discriminant Analysis and Application

Krishnaiah, P.R. and Kanal, L.N. (1982) Classification Pattern Recognition and Reduction of Dimensionality

Appendix

Figure1. Linear Regression

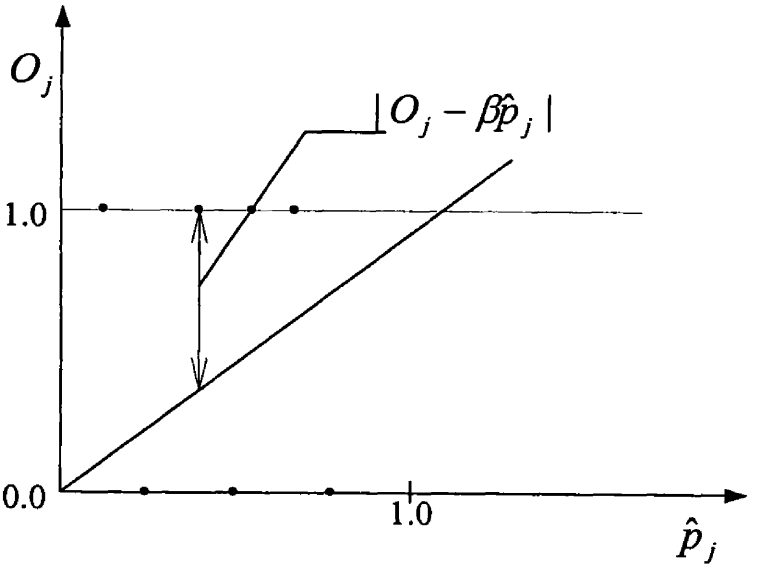


Figure 2. The true accuracy from k-NN and LDA.

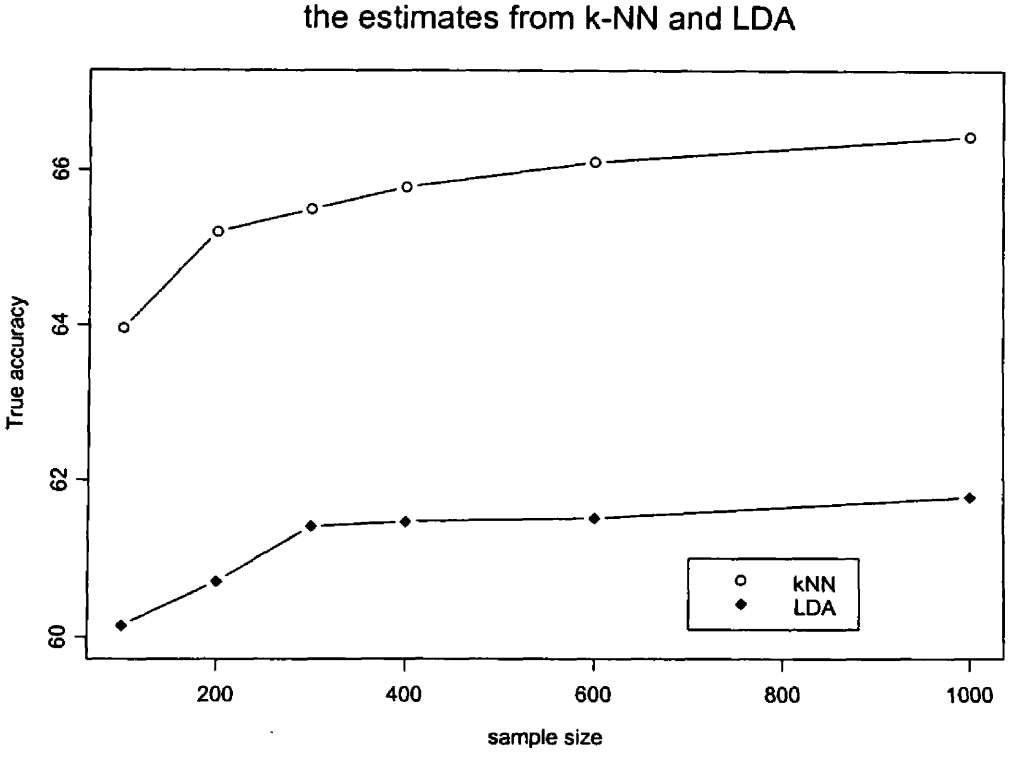


Table 1. LDA Sample size = 100 True acc for population = 60.17

	Means				Bias				RMSE			
k	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic
5	59.785	59.715	60.227	59.515	0.385	0.454	-0.057	0.655	5.695	3.838	5.199	5.346
10	60.027	59.501	60.467	59.512	0.143	0.669	-0.297	0.657	5.676	3.92	5.183	5.351
100	60.205	59.362	60.671	59.494	-0.035	0.808	-0.502	0.676	5.423	3.974	4.98	5.311

Table 2. LDA Sample size = 200 True acc for population = 61.18

	Means				Bias				RMSE			
k	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic
5	61.285	58.837	61.727	59.31	-0.101	2.347	-0.543	1.874	3.965	3.677	3.632	4.253
10	61.292	58.732	61.772	59.249	-0.107	2.452	-0.588	1.936	3.859	3.768	3.539	4.286
200	61.43	58.653	61.888	59.219	-0.246	2.531	-0.703	1.965	3.811	3.824	3.526	4.292

Table 3. LDA Sample size = 300 True acc for population = 61.41

	Means				Bias				RMSE			
k	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic
5	61.36	58.566	61.905	59.078	0.049	2.842	-0.497	2.331	2.898	3.61	2.668	3.74
10	61.515	58.491	62.043	59.056	-0.106	2.918	-0.634	2.352	2.89	3.663	2.667	3.742
300	61.53	58.426	62.077	59.006	-0.121	2.982	-0.669	2.403	2.855	3.719	2.651	3.777

Table 4. LDA Sample size = 400 True acc for population = 61.47

	Means				Bias				RMSE			
k	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic
5	61.225	58.179	61.849	58.666	0.246	3.292	-0.378	2.805	2.459	3.796	2.282	3.757
10	61.312	58.121	61.925	58.643	0.159	3.35	-0.454	2.829	2.511	3.839	2.343	3.793
400	61.337	58.085	61.954	58.621	0.134	3.386	-0.483	2.85	2.562	3.874	2.369	3.818

Table 5. LDA Sample size = 600 True acc for population = 61.51

	Means				Bias				RMSE			
k	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic
5	61.612	58.363	62.259	58.977	-0.098	3.15	-0.746	2.536	2.051	3.546	1.971	3.319
10	61.47	58.325	62.168	58.908	0.043	3.188	-0.654	2.606	1.986	3.582	1.91	3.357
600	61.663	58.302	62.327	58.939	-0.15	3.211	-0.813	2.575	2.052	3.601	1.996	3.344

Table 6. LDA Sample size = 1000 True acc for population = 61.87

	Means				Bias				RMSE			
k	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic
5	61.833	57.752	62.31	58.359	0.036	4.118	-0.44	3.51	1.562	4.312	1.483	3.887
10	61.847	57.731	62.32	58.349	0.023	4.139	-0.45	3.52	1.671	4.336	1.583	3.924
1000	61.827	57.712	62.32	58.329	0.043	4.157	-0.45	3.54	1.626	4.352	1.542	3.934

Table 7. k-NN Sample size = 100 True acc for population = 63.957

	Means				Bias				RMSE			
k	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic
5	63.41	64.59	64.77	63.93	0.55	−0.63	−0.82	0.03	5.95	3.10	5.33	5.74
10	63.95	64.64	64.56	64.83	0.13	−0.56	−0.48	−0.75	5.39	2.89	4.70	5.58
100	63.91	64.49	63.92	65.28	0.03	−0.55	0.02	−1.34	5.54	2.96	4.67	6.00

Table 8. k-NN Sample size = 200 True acc for population = 65.201

	Means				Bias				RMSE			
k	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic
5	64.90	67.34	65.56	67.90	0.30	-2.14	-0.36	-2.70	3.70	3.00	3.24	4.88
10	64.75	67.19	65.07	67.99	0.37	−2.06	0.06	−2.86	3.73	2.90	3.19	4.91
200	64.92	67.31	64.93	68.52	0.26	−2.13	0.24	−3.34	3.76	2.96	3.17	5.20

Table 9. k-NN Sample size = 300 True acc for population = 65.494

	Means				Bias				RMSE			
k	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic
5	65.16	68.31	65.66	69.12	0.33	-2.82	-0.17	-3.62	3.14	3.31	2.74	4.94
10	65.29	68.31	65.51	69.42	0.28	-2.73	0.06	-3.84	3.06	3.23	2.66	5.08
300	65.35	68.24	65.33	69.60	0.17	-2.72	0.19	-4.08	2.99	3.19	2.46	5.21

Table 10. k-NN Sample size = 400 True acc for population = 65.778

	Means				Bias				RMSE			
k	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic
5	65.3	68.73	65.64	69.72	0.48	−2.95	0.14	−3.94	2.79	3.31	2.38	4.93
10	65.53	68.78	65.64	70.02	0.24	−3.01	0.13	−4.25	2.66	3.34	2.25	5.11
400	65.49	68.79	65.50	70.16	0.33	−2.96	0.33	−4.34	2.69	3.29	2.22	5.16

Table 11. k-NN Sample size = 600 True acc for population = 66.097

	Means				Bias				RMSE			
k	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic
5	65.70	69.38	65.92	70.61	0.40	−3.28	0.18	−4.51	2.10	3.49	1.81	5.02
10	65.73	69.32	65.79	70.73	0.33	−3.26	0.27	−4.67	2.14	3.47	1.80	5.17
600	65.83	69.38	65.79	70.90	0.29	−3.26	0.33	−4.78	2.12	3.47	1.78	5.28

Table 12. k-NN Sample size = 1000 True acc for population = 66.428

	Means				Bias				RMSE			
k	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic	CV	Max Post	Linear	Logistic
5	65.86	69.84	66.00	71.18	0.57	−3.41	0.43	−4.75	1.78	3.54	1.52	5.07
10	65.90	69.83	65.93	71.31	0.55	−3.38	0.52	−4.85	1.68	3.50	1.45	5.13
1000	65.75	69.76	65.75	71.23	0.66	−3.34	0.66	−4.82	1.83	3.47	1.56	5.15

Figure 3. LDA Bias for 5-fold Cross-Validation.

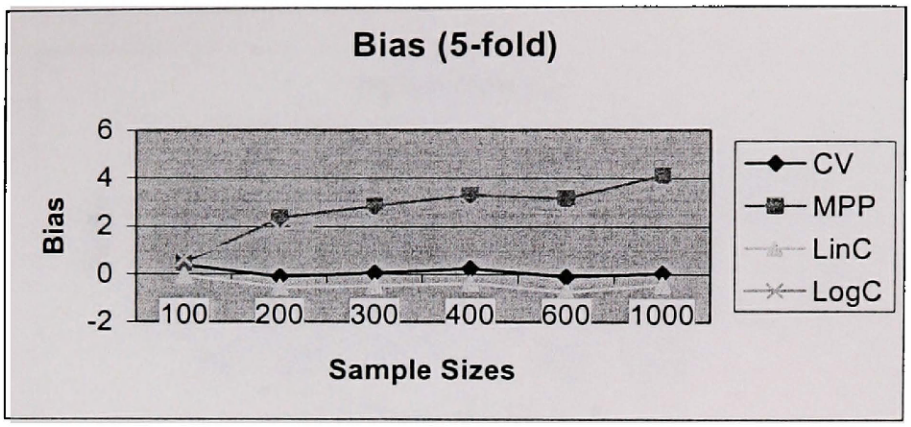


Figure 4. LDA Bias for 10-fold Cross-Validation.

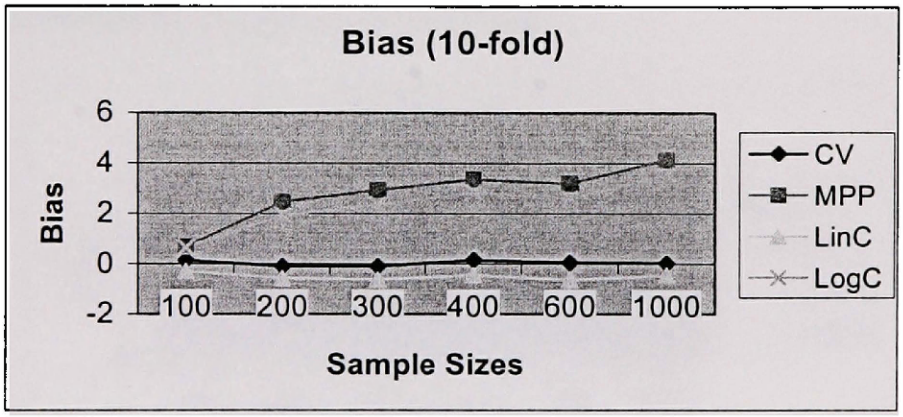


Figure 5. LDA Bias for n-fold Cross-Validation.

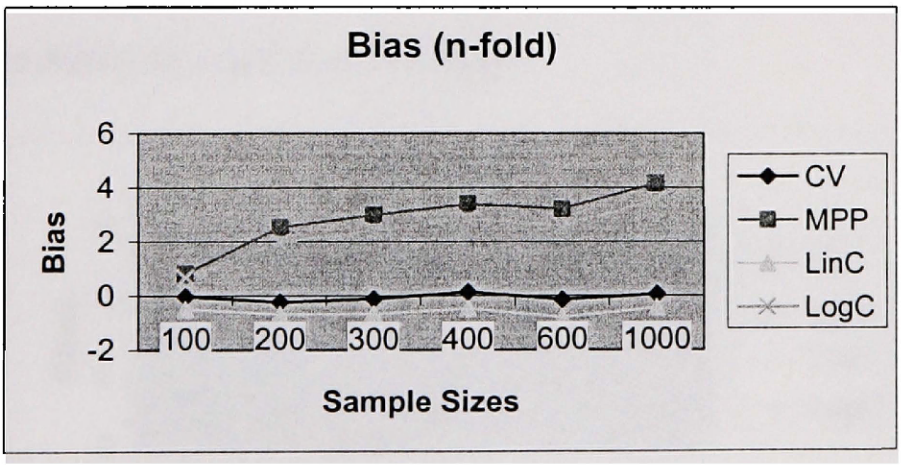


Figure 6. LDA RMSE for 5-fold Cross-Validation.

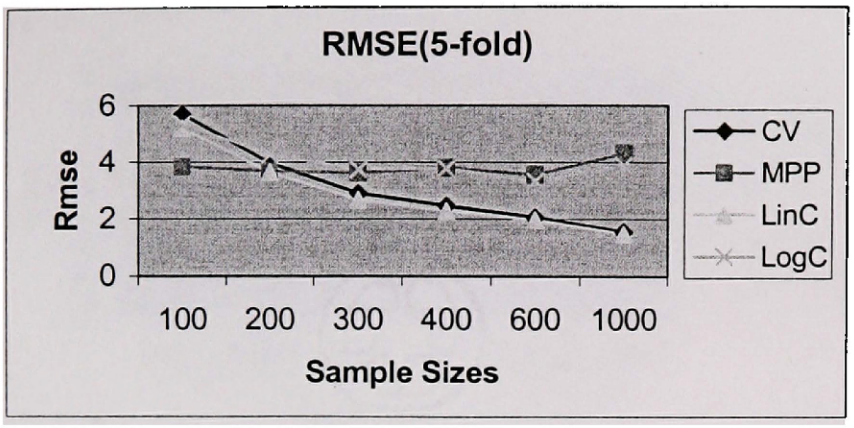


Figure 7. LDA RMSE for 10-fold Cross-Validation.

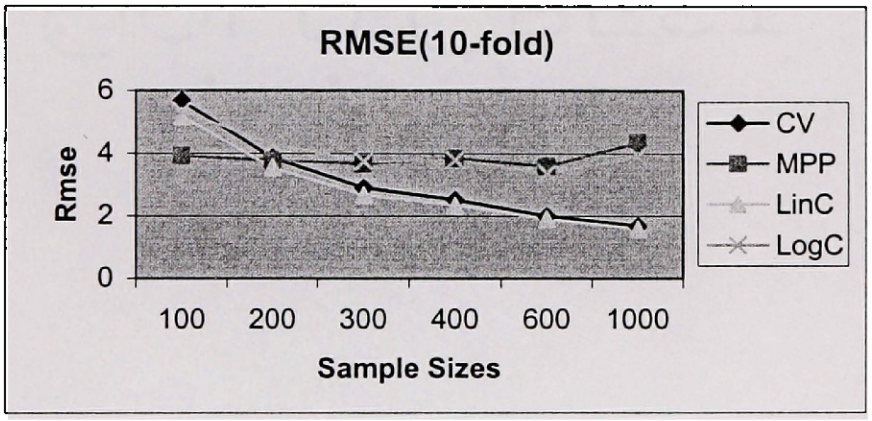


Figure 8. LDA RMSE for n-fold Cross-Validation.

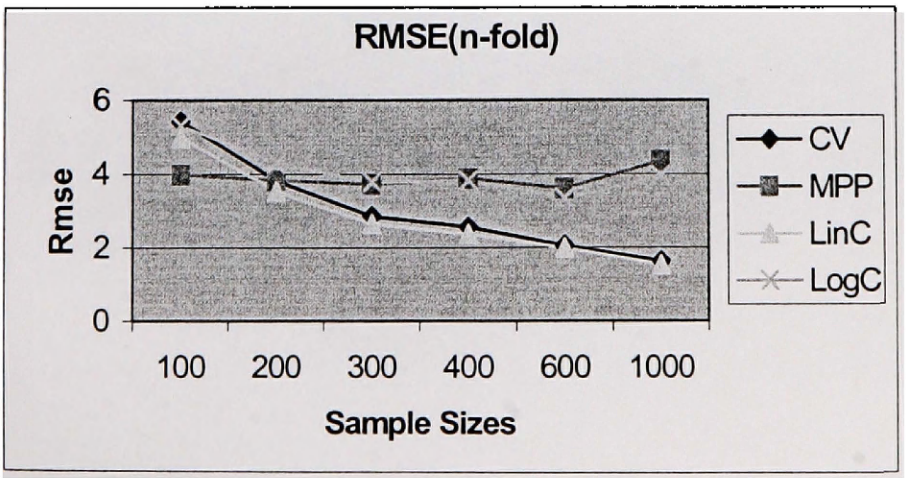


Figure 9. *k*-NN Bias for 5-fold Cross-Validation.

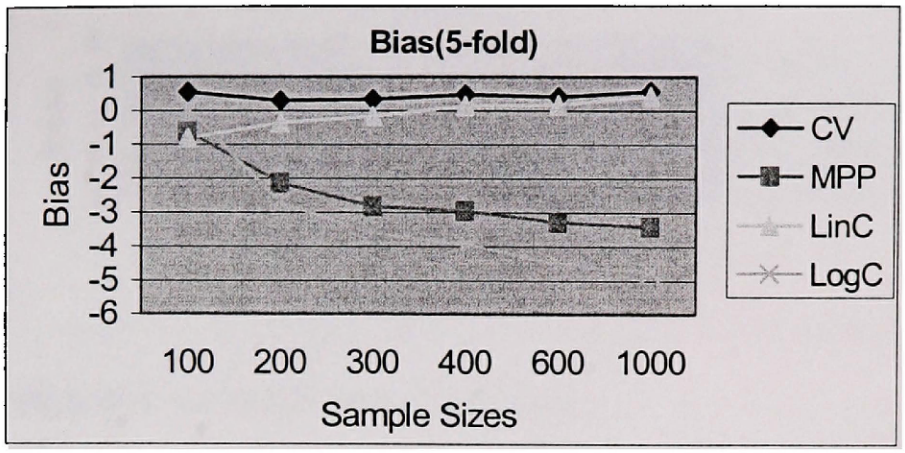


Figure 10. *k*-NN Bias for 10-fold Cross-Validation.

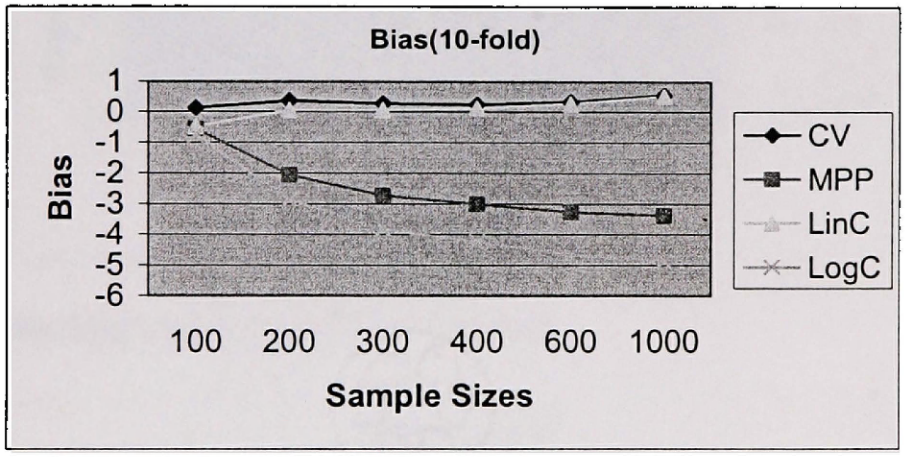


Figure 11. *k*-NN Bias for n-fold Cross-Validation.

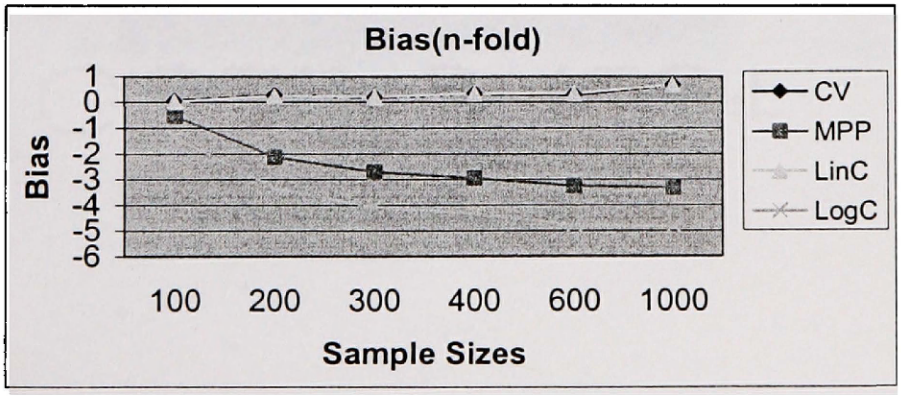


Figure 12. *k*-NN RMSE for 5-fold Cross-Validation.

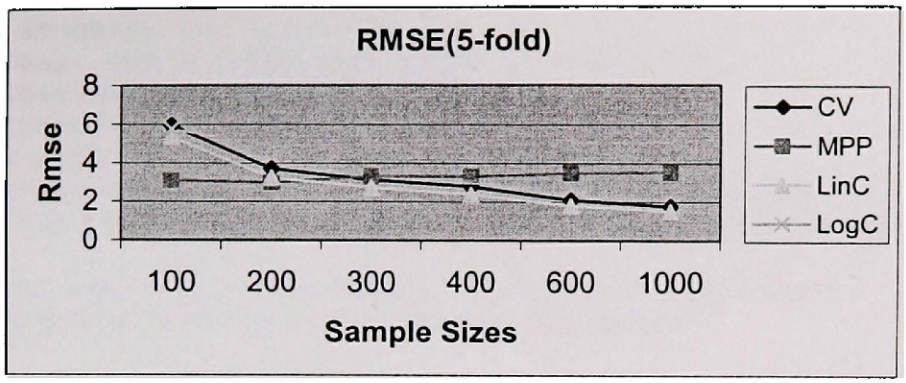


Figure 13. *k*-NN RMSE for 10-fold Cross-Validation.

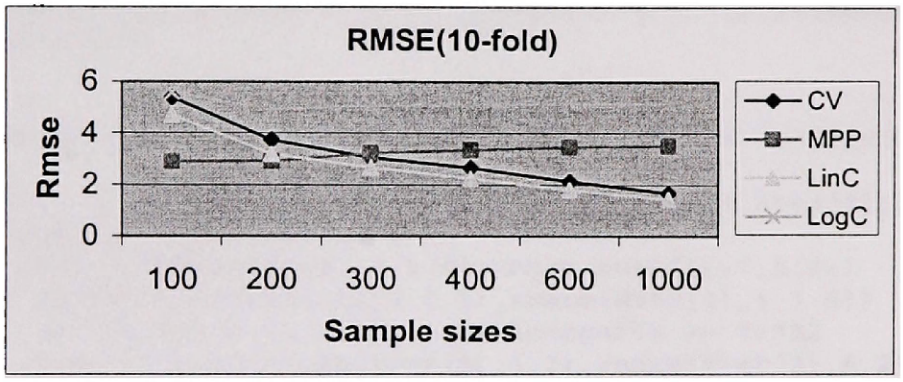
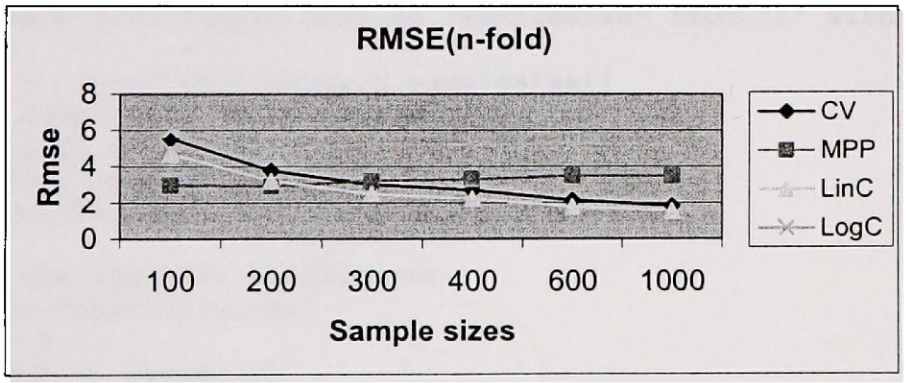


Figure 14. *k*-NN RMSE for n-fold Cross-Validation.



Splus code:

```
#=========================================================
#Simulate.ssc  Aug,2003
#This code is going to calculate
#true accuracy, the average Mean, Bias and Rmse for
#Cross Validation, Max Posterior Prob,
#Linear Calibration, Logistic Calibration estimates
#for k-fold with the population size n=10,000,
#sample size as samplesizes<-c(100,200,300,400,600,1000),
#for each size, repeat 1000 times by using Linear Discriminant
#Classifier.
#=========================================================
t <- options(width=120,digits=4,compact=1e5,echo=F)

#Generate a sample with sample size as "n" from 3 populations
GEN <- function(n){
      r <- runif(n,0,1)

      l <- as.integer(r<1/3) + 2*as.integer(r>1/3)*as.integer(r<2/3) +
3*as.integer(r>2/3)

      nvec <-
c(sum(as.integer(l==1)),sum(as.integer(l==2)),sum(as.integer(l==3)))

      group <- c(rep(1,nvec[1]),rep(2,nvec[2]),rep(3,nvec[3]))
      z <- runif(nvec[1],0,1)<0.6
      X1 <-cbind(rnorm(nvec[1],7,1.5),rnorm(nvec[1],7,1.5))
      X2 <- cbind(rnorm(nvec[1],3,1.5),rnorm(nvec[1],3,1.5))
      X <-  as.integer(z == F)*X1 + as.integer(z == T)*X2
      X <- rbind(X,cbind(rnorm(nvec[2],4,2),rnorm(nvec[2],6,2)))
      X <- rbind(X,cbind(rnorm(nvec[3],7.5,3),rnorm(nvec[3],3.5,3)))

      return(group,nvec,X)}

#Draw a sample with sample size as "samplesize" from "X" with sample
size as "n"
DrawSample <- function(X,group,n,samplesize){
      r <- sample(1:n, size=samplesize)
      TrainingX <- X[r,]
      TrainingGroup <- group[r]
      return(TrainingX,TrainingGroup)}

#Calculate the logistic coefficents
LogCalCoeff<-function(y,xx){
      ab <- 1
      ylength <- length(y)
      a <- solve(t(xx)%*%(xx),t(xx)%*%log((y + 0.1)/(1.1 - y)))
      m <- 1/(1+exp(-xx%*%a))
      w <- m*(1-m)
            while(ab > 1E-7){
            aold <- a
            a <- a + solve(t(xx)%*%(w*xx),t(xx)%*%(y - m))
            m <- 1/(1+exp(-xx%*%a))
            w <- m*(1-m)
           ab <- abs(a-aold)}
      return(a)          }
```

```
#Classify Xtest by using Training Group
Classify <- function(TrainingGroup,TrainingX,Xtest){
#the sample size in Training Group
      Num <- length(TrainingGroup)
#the size in each Group in the Training Group
      nvec <- c(sum(as.integer(TrainingGroup==1)),

sum(as.integer(TrainingGroup==2)),sum(as.integer(TrainingGroup==3)))
#the probability for each group
      pvec <- nvec/Num

#calculate the mean and variance for each group
      m1 <- colMeans(TrainingX[TrainingGroup==1,])
      s1 <- var(TrainingX[TrainingGroup==1,])
      m2 <- colMeans(TrainingX[TrainingGroup==2,])
      s2 <- var(TrainingX[TrainingGroup==2,])
      m3 <- colMeans(TrainingX[TrainingGroup==3,])
      s3 <- var(TrainingX[TrainingGroup==3,])

#calculate get the pooled variance matrix
      s <- ((nvec[1]-1)*s1+(nvec[2]-1)*s2+(nvec[3]-1)*s3)/(Num-3)

#calculate the probs for each observation for each group
      f1 <- pvec[1]*dmvnorm(Xtest,mean=m1,cov=s)
      f2 <- pvec[2]*dmvnorm(Xtest,mean=m2,cov=s)
      f3 <- pvec[3]*dmvnorm(Xtest,mean=m3,cov=s)
      denom <- f1 + f2 + f3
      posterior <- cbind(f1,f2,f3)/denom

#find the group with the largest prob
      maxprob <- apply(posterior,MARGIN=1,FUN=max)
      predgroup <- apply(posterior==maxprob,MARGIN=1,FUN=which)

      return(maxprob,predgroup)}

#Population size
n <- 10000

#Generate the random sample with sample size "n". "n" is large enough.
We regard this sample as the whole population.

data <- GEN(n)
group <- data$group
nvec <- data$nvec
X <- data$X

#calculate by using the following sample sizes
samplesizes <- c(100,200,300,400,600,1000)

#calculate 5-fold, 10-fold, samplesizes-fold cross validation.
kfold <- c(5,10,0)

#For each sample size, repeat "nsamples" times to get the average value.
nsamples <- 1000

MeansMatrix <- matrix(0,length(samplesizes),12)
```

```
BiasMatrix <-  matrix(0,length(samplesizes),12)
RMSEMatrix <-  matrix(0,length(samplesizes),12)

acc <- matrix(0,nsamples,13)

#Begin to calculate the 12 values with the "kone"th sample size
for(kone in 1:length(samplesizes)){
      samplesize <- samplesizes[kone]
      print(c("samplesize = ",samplesize),quote = F)

      sq <- 1:samplesize
      kfold[length(kfold)] <- samplesize

      predgroupn2<-rep(0,samplesize)
      maxprobn2<-rep(0,samplesize)

      for(i in 1:nsamples){
#calculate the true accuracy
            TrainingData <- DrawSample(X,group,n,samplesize)

            TrainingGroup <- TrainingData$TrainingGroup
            TrainingX <- TrainingData$TrainingX

            Results <- Classify(TrainingGroup, TrainingX , X)
            acc[i,1] <- 100*mean(Results$predgroup==group)
            print(c(dim(X)[1],"Sample size =",samplesize,"Sample =",i,
                  "Acc = ",acc[i,1]),quote = F)

            for(ktwo in 1:length(kfold)){

                  if (ktwo == length(kfold))    index <- sq
                  else index <-sample(rep(1:kfold[ktwo],
                   samplesize/kfold[ktwo]), size=samplesize)

                  for(j in 1:kfold[ktwo]){
                   if (ktwo == kfold[length(kfold)])  holdout <- j
                   else holdout <- sq[index==j]
                   heldin <- sq[index != j]

                        Y <- TrainingX[holdout,]
                        TrainingGroupnew <- TrainingGroup[heldin]
                        TrainingXnew <- TrainingX[heldin,]

                        Results <-
                         Classify(TrainingGroupnew,TrainingXnew,Y)

                        maxprobn2[holdout] <- Results$maxprob
                        predgroupn2[holdout] <- Results$predgroup
      }

#calculate the cross-validation accuracy estimate
            acc[i,(ktwo-1)*4+2] <- 100*mean(predgroupn2==TrainingGroup)
#calculate the max. posterior prob accuracy estimate
            acc[i,(ktwo-1)*4+3] <- 100*mean(maxprobn2)
#calculate the linear and logistic calibration accuracy estimates
            yy <- as.integer(predgroupn2 == TrainingGroup)
            bb <- sum(yy*maxprobn2)/sum(maxprobn2^2)
```

```
            aa <- LogCalCoeff(yy,maxprobn2)
            LogCalmaxprob <- 1/(1 + ((1 - maxprobn2)/maxprobn2)^aa)

            acc[i,(ktwo-1)*4+4] <- bb*acc[i,(ktwo-1)*4+3]
            acc[i,(ktwo-1)*4+5] <- 100*mean(LogCalmaxprob)
            }
      }

#calculate the average value for Mean, Bias, Rmse
      MeansMatrix[kone,] <- colMeans(acc[,2:13])
      M <- mean(acc[,1])
      BiasMatrix[kone,] <- colMeans(M - acc[,2:13])
      RMSEMatrix[kone,] <- sqrt(colMeans((M - acc[,2:13])^2))

#print them out
      print(c("Sample size = ",samplesizes[kone]),quote = F)
      print(c("True acc for pop'n = ",round(M,2)),quote=F)

      PlotterMeans <-
        rbind(MeansMatrix[kone,1:4],MeansMatrix[kone,5:8],
        MeansMatrix[kone,9:12])
      PlotterBias <-
        rbind(BiasMatrix[kone,1:4],BiasMatrix[kone,5:8],
        BiasMatrix[kone,9:12])
      PlotterRMSE <-
        rbind(RMSEMatrix[kone,1:4],RMSEMatrix[kone,5:8],
        RMSEMatrix[kone,9:12])

      cv <- c("-fold",kfold[1:(length(kfold)-1)],samplesizes[kone])


      print("Means",quote = F)
      print(cbind(cv,rbind(c("CV","Max Post","Linear","Logistic"),
      round(PlotterMeans,dig=3))),quote=F)

      print("Bias",quote = F)
      print(cbind(cv,rbind(c("CV","Max Post","Linear","Logistic"),
      round(PlotterBias,dig=3))),quote=F)

      print("RMSE",quote = F)
      print(cbind(cv,rbind(c("CV","Max Post","Linear","Logistic"),
      round(PlotterRMSE,dig=3))),quote=F)
}
```

Splus code:

```
#=========================================================
#Simulate.ssc  Aug,2003
#This code is going to calculate
#true accuracy, the average Mean, Bias and Rmse for
#Cross Validation, Max Posterior Prob,
#Linear Calibration, Logistic Calibration estimates
#for k-fold with the population size n=10,000,
#sample size as samplesizes<-c(100,200,300,400,600,1000),
#for each size, repeat 1000 times by using kNN
#Classifier.
#=========================================================
t <- options(width=120,digits=4,compact=1e5,echo=F)

#Generate a sample with sample size as "n" from 3 populations
GEN <- function(n){

      r <- runif(n,0,1)
      l <- as.integer(r<1/3) + 2*as.integer(r>1/3)*as.integer(r<2/3) +
3*as.integer(r>2/3)
      nvec <-
c(sum(as.integer(l==1)),sum(as.integer(l==2)),sum(as.integer(l==3)))
      group <- c(rep(1,nvec[1]),rep(2,nvec[2]),rep(3,nvec[3]))

      z <- runif(nvec[1],0,1)<0.6
      X1 <-cbind(rnorm(nvec[1],7,1.5),rnorm(nvec[1],7,1.5))
      X2 <- cbind(rnorm(nvec[1],3,1.5),rnorm(nvec[1],3,1.5))
      X <-  as.integer(z == F)*X1 + as.integer(z == T)*X2

      X <- rbind(X,cbind(rnorm(nvec[2],4,2),rnorm(nvec[2],6,2)))

      X <- rbind(X,cbind(rnorm(nvec[3],7.5,3),rnorm(nvec[3],3.5,3)))
      return(group,nvec,X)}

#Draw a sample with sample size as "samplesize" from "X" with sample
size as "n"
DrawSample <- function(X,group,n,samplesize){
      r <- sample(1:n, size=samplesize)
      TrainingX <- X[r,]
      TrainingGroup <- group[r]
      return(TrainingX,TrainingGroup)}

#Calculate the logistic coefficents
LogCalCoeff<-function(y,xx){
      ab <- 1
      ylength <- length(y)
      a <- solve(t(xx)%*%(xx),t(xx)%*%log((y + 0.1)/(1.1 - y)))
      m <- 1/(1+exp(-xx%*%a))
      w <- m*(1-m)

            while(ab > 1E-7){
            aold <- a
            a <- a + solve(t(xx)%*%(w*xx),t(xx)%*%(y - m))
            m <- 1/(1+exp(-xx%*%a))
            w <- m*(1-m)
```

```
        ab <- abs(a-aold)}
        return(a)            }

ClassifykNN <- function(TrainingGroup,TrainingX,Xtest, Ind){
            kNN<-10
            Numtr<-dim(TrainingX)[1]
       if(Ind == 1) Num<-1
      else  Num<-dim(Xtest)[1]
#get unique groups
       labels<-unique(sort(TrainingGroup))
       g<-length(labels)
       prob<-rep(0,g)
       posterior<-matrix(0,Num,g)

       r<-1

#calculate distance between rth observation in the test sample and all
the observations in training sample
       while(r<=Num){

             if(Num ==1) t0<-Xtest
           else  t0<-Xtest[r,]

               M<-as.matrix(rep(1,Numtr))

              M<-M%*%t0

                 D<-rowSums((TrainingX-M)^2)
              index<-sort.list(D)

#calculate how many observations belong to each group
              i<-1
              while(g+1>i) {prob[i]<-
sum(as.integer(TrainingGroup[index[1:kNN]]==labels[i]))
                   i<-i+1}

#break the ties by increasing the number of neighborhood
             kNNr <- kNN+1
             while(sum(as.integer(prob==max(prob)))>1)       {
               i<-1
               while(g+1>i) {prob[i]<-
sum(as.integer(TrainingGroup[index[1:kNNr]]==labels[i]))
                  i<-i+1}
            kNNr <- kNNr + 1}

            posterior[r,1:g]<-prob[1:g]/sum(prob)

            r<-r+1}
             maxprob <- apply(posterior,MARGIN=1,FUN=max)
                predgroup <-
apply(posterior==maxprob,MARGIN=1,FUN=which)

                return(maxprob,predgroup)
      }


#Population size
```

```
n <- 10000

#Generate the random sample with sample size "n". "n" is large enough.
We regard this sample as the whole population.
data <- GEN(n)
group <- data$group
nvec <- data$nvec
X <- data$X

#calculate by using the following sample sizes
samplesizes <- c(100,200,300,400,600,1000)

#calculate 5-fold, 10-fold, samplesizes-fold cross validation.
kfold <- c(5,10,0)

#For each sample size, repeat "nsamples" times to get the average
value.
nsamples <- 1000

MeansMatrix <- matrix(0,length(samplesizes),12)
BiasMatrix <-  matrix(0,length(samplesizes),12)
RMSEMatrix <-  matrix(0,length(samplesizes),12)

acc <- matrix(0,nsamples,13)

#Begin to calculate the 12 values with the "kone"th sample size
for(kone in 1:length(samplesizes)){
      samplesize <- samplesizes[kone]
      print(c("samplesize = ",samplesize),quote = F)

      sq <- 1:samplesize
      kfold[length(kfold)] <- samplesize

      predgroupn2<-rep(0,samplesize)
      maxprobn2<-rep(0,samplesize)

      for(i in 1:nsamples){
#calculate the true accuracy
            TrainingData <- DrawSample(X,group,n,samplesize)

            TrainingGroup <- TrainingData$TrainingGroup
            TrainingX <- TrainingData$TrainingX

            Results <- ClassifykNN(TrainingGroup, TrainingX , X, 0)
            acc[i,1] <- 100*mean(Results$predgroup==group)

            print(c(dim(X)[1],"Sample size =",samplesize,"Sample =",i,
                  "Acc = ",acc[i,1]),quote = F)

            for(ktwo in 1:length(kfold)){

             Ind<-0

                  if (ktwo == length(kfold))    index <- sq
                  else index <-
sample(rep(1:kfold[ktwo],samplesize/kfold[ktwo]), size=samplesize)
```

```
                    for(j in 1:kfold[ktwo]){
                     if (ktwo == length(kfold))    {
                    holdout <- j
                     Ind<-1}
                     else holdout <- sq[index==j]
                          heldin <- sq[index != j]

                          Y <- TrainingX[holdout,]
                          TrainingGroupnew <- TrainingGroup[heldin]
                          TrainingXnew <- TrainingX[heldin,]

                          Results <-
ClassifykNN(TrainingGroupnew,TrainingXnew,Y,Ind)

                          maxprobn2[holdout] <- Results$maxprob
                          predgroupn2[holdout] <- Results$predgroup
      }
#calculate the cross-validation accuracy estimate
            acc[i,(ktwo-1)*4+2] <- 100*mean(predgroupn2==TrainingGroup)
#calculate the max. posterior prob accuracy estimate
            acc[i,(ktwo-1)*4+3] <- 100*mean(maxprobn2)
 #calculate the linear calibration and logistic calibration accuracy
estimate
            yy <- as.integer(predgroupn2 == TrainingGroup)
            bb <- sum(yy*maxprobn2)/sum(maxprobn2^2)

      aa <- LogCalCoeff(yy,maxprobn2)
            LogCalmaxprob <- 1/(1 + ((1 - maxprobn2)/maxprobn2)^aa)

            acc[i,(ktwo-1)*4+4] <- bb*acc[i,(ktwo-1)*4+3]
            acc[i,(ktwo-1)*4+5] <- 100*mean(LogCalmaxprob)
            }
      }
#calculate the average value for Mean, Bias, Rmse
      MeansMatrix[kone,] <- colMeans(acc[,2:13])
      M <- mean(acc[,1])
      BiasMatrix[kone,] <- colMeans(M - acc[,2:13])
      RMSEMatrix[kone,] <- sqrt(colMeans((M - acc[,2:13])^2))

#print them out
      print(c("Sample size = ",samplesizes[kone]),quote = F)
      print(c("True acc for pop'n = ",round(M,2)),quote=F)

      PlotterMeans <-
rbind(MeansMatrix[kone,1:4],MeansMatrix[kone,5:8],MeansMatrix[kone,9:12
])
      PlotterBias <-
rbind(BiasMatrix[kone,1:4],BiasMatrix[kone,5:8],BiasMatrix[kone,9:12])
      PlotterRMSE <-
rbind(RMSEMatrix[kone,1:4],RMSEMatrix[kone,5:8],RMSEMatrix[kone,9:12])

      cv <- c("-fold",kfold[1:(length(kfold)-1)],samplesizes[kone])
      print("Means",quote = F)
      print(cbind(cv,rbind(c("CV","Max
Post","Linear","Logistic"),round(PlotterMeans,dig=3))),quote=F)
      print("Bias",quote = F)
```

```
      print(cbind(cv,rbind(c("CV","Max
Post","Linear","Logistic"),round(PlotterBias,dig=3))),quote=F)
      print("RMSE",quote = F)
      print(cbind(cv,rbind(c("CV","Max
Post","Linear","Logistic"),round(PlotterRMSE,dig=3))),quote=F)
}
```