STAE03: Business Analytics

Assignment 3

Diana (990820T222)

1. Introduction

This assignment is divided into two parts. The first part utilizes the MBA data which provide admission data, GPA, and GMAT of applicants to a graduate school in business. It aims to predict the probability of the decision of the admission with Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA). Since GPA and GMAT have different range of values, thus one standardized the data first and proceed to the analysis of the data. The dataset contains 85 entries and one divided the dataset into training and test data with a proportion of 70:15 = 14:3.

The second part utilizes the Default data from ISLR library which provide default data, annual income, monthly credit card balance, and if the person is a student or not. It aims to predict whether a person will default on his or her credit card with LDA, QDA, K-Nearest Neighbours (KNN) and Logistic Regression. The predictor variables are monthly credit card balance (*balance*) and if the person is a student or not (*student*). The dataset contains 10000 entries and one divided the dataset into training and test data with a proportion of 9000:1000 = 9:1.

2. Part 1: MBA data

2.1. Linear Discriminant Analysis (LDA)

One fitted LDA model to the standardized MBA training data and obtained the following results:

```
Call:
lda(mba.Decision ~ std_gpa + std_gmat, data = std_mba, subset = train)

Prior probabilities of groups:
    admit    border  notadmit
0.3714286 0.3285714 0.3000000

Group means:
             std_gpa   std_gmat
admit     0.95219835  0.8192074
border    0.07342438 -0.5745463
notadmit -1.07963790 -0.4733661

Coefficients of linear discriminants:
               LD1        LD2
std_gpa  2.1901518  0.8031497
std_gmat 0.7101502 -1.1660983

Proportion of trace:
  LD1   LD2
0.958 0.042
```

Figure 2.1.1: Linear Discriminant Analysis result

ldatestclass	admit	border	notadmit
admit	5	0	0
border	0	3	1
notadmit	0	0	6

Figure 2.1.2: LDA Prediction vs Actual result of admission

```
> mean(ldatestclass == test$mba.Decision)
[1] 0.9333333
```

Figure 2.1.3: LDA Proportion of correct classification

The test error is only 1/15 = 6.67%. It is low because there is only 1 error out of 15 observation.

2.2. Quadratic Discriminant Analysis (QDA)

One fitted QDA model to the standardized MBA training data and obtained the following results:

```
Call:
qda(mba.Decision ~ std_gpa + std_gmat, data = std_mba, subset = train)

Prior probabilities of groups:
    admit    border  notadmit
0.3714286 0.3285714 0.3000000

Group means:
              std_gpa   std_gmat
admit      0.95219835  0.8192074
border     0.07342438 -0.5745463
notadmit -1.07963790 -0.4733661
```

Figure 2.2.1: Quadratic Discriminant Analysis result

qdatestclass	admit	border	notadmit
admit	5	0	0
border	0	3	0
notadmit	0	0	7

Figure 2.2.2: QDA Prediction vs Actual result of admission

```
> mean(qdatestclass == test$mba.Decision)
[1] 1
```

Figure 2.2.3: QDA Proportion of correct classification

The test error is 0% or in other words, QDA predicts the result of admission 100% accurately for all 15 observation.

2.3. Comparison of LDA and QDA

From the results above, one could conclude that QDA is a better fit to the data based on its proportion of correct classification which is 100%. However, LDA also performs pretty well since its test accuracy is 93.33%, which only differs 1 error from QDA. The performance of both model are extremely good and it might lead to a biased conclusion. Note that the number of observation in the test data is only

15. One assumed that low number of observation could potentially create high bias towards the interpretation of the performance of the model. For instance, one observed that there is a 6.67% drop in proportion of correct classification using LDA. 6.67% is relatively high if the number of observation is high as well. The number is quite misleading because one could easily conclude that QDA is more superior than LDA for this data although in fact, they perform relatively the same as their test error only differs by 1 observation. To further analyse the performance of both models, one could also measure the proportion of correct classification on the training data. The results are depicted in figure 2.3.1 and 2.3.2:

ldatrainclass	admit	border	notadmit
admit	23	1	0
border	3	21	1
notadmit	0	1	20

Figure 2.3.1: LDA Prediction vs Actual result of admission on training data

qdatrainclass	admit	border	notadmit
admit	25	1	0
border	1	22	1
notadmit	0	0	20

Figure 2.3.2: QDA Prediction vs Actual result of admission on training data

The proportion of correct classification on the training data using LDA and QDA are 91.43% and 95.71% respectively. These results further support the conclusion that QDA is a better fit to the data but not that much better than LDA.

3. Part 2: Default data

3.1. Linear Discriminant Analysis (LDA)

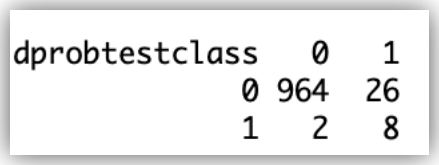
```
Call:
lda(defaultn ~ studentn + balancer, data = Default, subset = train)

Prior probabilities of groups:
         0          1
0.96677778 0.03322222

Group means:
   studentn  balancer
0 0.2913458 0.8020951
1 0.3913043 1.7479952

Coefficients of linear discriminants:
                LD1
studentn -0.2142023
balancer  2.2430955
```

Figure 3.1.1: Linear Discriminant Analysis result



dprobtestclass	0	1
0	964	26
1	2	8

Figure 3.1.2: LDA Prediction vs Actual result on test data

```
> mean(dprobtestclass == test$defaultn)
[1] 0.972
```

Figure 3.1.3: LDA proportion of correct classification

The proportion of correct classification with cut-off point = 0 .5 is 0.972 which one considered as to be pretty high. However, its sensitivity is only 0.2353 and its specificity is 0.9979. This indicates that the model is predicting most of the non-default cases almost accurately but it performs poorly in predicting the default cases. In this test data, only 8 default observation are accurately predicted. This suggests that there should be an improvement in terms of sensitivity and specificity by adjusting the cut-off point. One obtained the optimal cut-off point as follows:

	cutoff	total	sensitivity	specificity
39	0.039	1.762222	0.90301	0.8592116

Figure 3.1.4: LDA Optimal Cut-off point

With this new cut-off point, sensitivity drops to 0.90301 but there is a significant improvement on the specificity value which rises to 0.8592. However, the proportion also drops to 0.861.

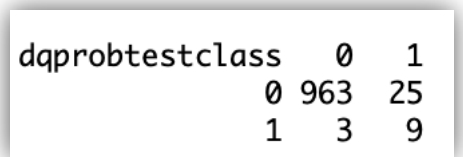
3.2. Quadratic Discriminant Analysis (QDA)

```
qda(defaultn ~ studentn + balancer, data = Default, subset = train)

Prior probabilities of groups:
         0          1
0.96677778 0.03322222

Group means:
   studentn  balancer
0 0.2913458 0.8020951
1 0.3913043 1.7479952
```

Figure 3.2.1: Quadratic Discriminant Analysis result



dqprobtestclass	0	1
0	963	25
1	3	9

Figure 3.2.2: QDA Prediction vs Actual result on test data

```
> mean(dqprobtestclass == test$defaultn)
[1] 0.972
```

Figure 3.2.3: QDA proportion of correct classification

QDA performs approximately similar to LDA for this dataset. When cut-off point = 0.5, sensitivity and specificity are 0.2647 and 0.9969 respectively. This indicates that the model is predicting most of the non-default cases almost accurately but it performs poorly in predicting the default cases. In this test data, only 9 default observation are accurately predicted. Nevertheless, it predicts 97.2% cases accurately. In order to improve the sensitivity of this model, one needed to adjust the cut-off point to its optimal value which is pretty similar with LDA's optimal cut-off point. However, it has lower sensitivity but higher specificity than LDA. With the optimal cut-off point, specificity rises from 0.36 to 0.86 and sensitivity drops quite significantly to 0.8997 and the proportion of correct classification also decreases to 86.5%.

	cutoff	total	sensitivity	specificity
38	0.038	1.76267	0.8996656	0.8630043

Figure 3.2.4: QDA optimal cut-off point

3.3. K-nearest Neighbours (KNN)

One trained the dataset by utilizing KNN and obtained the test result as follows:

	pcorrn1	pcorrn3	pcorrn5	pcorrn7	pcorrn9	pcorrn11	pcorrn13	pcorrn15
1	0.948	0.967	0.967	0.965	0.967	0.964	0.967	0.968

Figure 3.3.1: Proportion of correct classification result

By iterating through odd numbers from 1 to 15 as the value of K, one could observe that the proportion of correct classification results are roughly similar. It seems that K = 15 gives the highest proportion. One ran a cross-validation to validate this assumption.

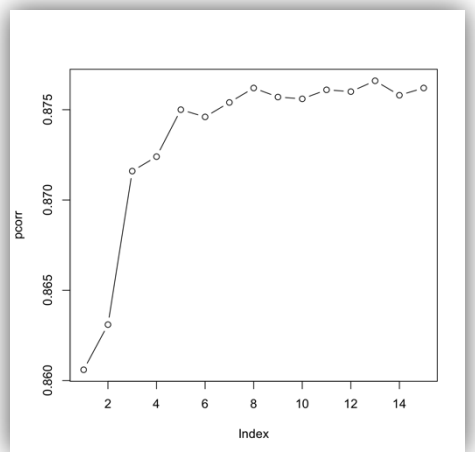


Figure 3.3.2: Cross-validation result

From the cross-validation result, one noticed that K = 15 does not give the highest proportion. It seems that K = 13 yields the highest proportion of correct classification. Currently, one has two candidates of the values of K that could potentially give the most accurate result in predicting default. One learned that when K = 13, sensitivity and specificity are 0.3824 and 0.9876 respectively. In addition, when K = 15, sensitivity and specificity are 0.3824 and 0.9886 respectively. Thus, by determining the results of cross-validation, sensitivity and specificity, one concluded that KNN performs the best in this data when K = 15.

```
nearest13   0   1
        0 954  21
        1  12  13
```

Figure 3.3.3: Prediction vs Actual result for K = 13.

```
nearest15   0   1
        0 955  21
        1  11  13
```

Figure 3.3.4: Prediction vs Actual result for K = 15

3.4. Logistic Regression

One trained the dataset by utilizing Logistic Regression and obtained the results as follows:

```
Call:
glm(formula = defaultn ~ studentn + balancer, family = binomial,
    data = Default, subset = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4675  -0.1413  -0.0552  -0.0199   3.7524

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -10.8050     0.3917 -27.583  < 2e-16 ***
studentn     -0.6677     0.1554  -4.296 1.74e-05 ***
balancer      5.7719     0.2464  23.428  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2623.9  on 8999  degrees of freedom
Residual deviance: 1405.9  on 8997  degrees of freedom
AIC: 1411.9

Number of Fisher Scoring iterations: 8
```

Figure 3.4.1: Logistic regression result

```
Single term deletions

Model:
defaultn ~ studentn + balancer
         Df Deviance    AIC     LRT  Pr(>Chi)
<none>        1405.9 1411.9
studentn  1   1425.2 1429.2   19.37 1.078e-05 ***
balancer  1   2610.7 2614.7 1204.79 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 3.4.2: Likelihood Ratio Test (LRT) result

By fitting Logistic Regression model to the dataset, one could understand further the relationship between the predictor variables and the response variable. Furthermore, one learned which predictor variables are significant in predicting the response variable. As observed from the results above, the variable student seems to have a negative relationship with default. It might indicate that if a person is a student, the probability for him/her to default is most likely to be lower than if a person is not a student. Moreover, from the Wald's test and Likelihood Ratio Test, one concluded that both predictor variables are statistically significant.

When cut-off point = 0.5, one obtained the following result:

```
   Cell Contents
|-------------------------|
|                   Count |
|             Row Percent |
|-------------------------|

Total Observations in Table:  1000

               | dtest$pred.c
dtest$defaultn |       No  |      Yes  | Row Total |
---------------|-----------|-----------|-----------|
             0 |      959  |        7  |      966  |
               |   99.275% |    0.725% |   96.600% |
---------------|-----------|-----------|-----------|
             1 |       22  |       12  |       34  |
               |   64.706% |   35.294% |    3.400% |
---------------|-----------|-----------|-----------|
  Column Total |      981  |       19  |     1000  |
---------------|-----------|-----------|-----------|
```

Figure 3.4.5: Actual result vs Prediction result when cut-off = 0.5

The percentage of correct classification is 0.971, so its test error is 0.029. Its sensitivity and specificity are 35.294% and 99.275% when cut-off point = 0.5. In order to look for the optimal sensitivity and specificity, one observed the result from the ROC-curve in figure 3.4.6. The result is quite disappointing

as there is no combination of sensitivity and specificity that is closer to the upper left corner of the graph.

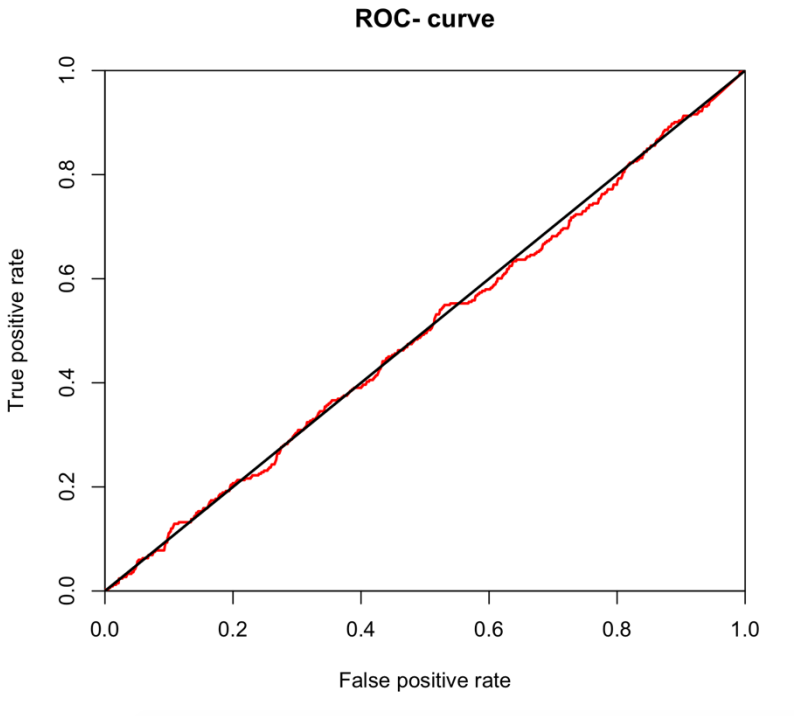


Figure 3.4.6: ROC-curve

However, one analysed that as the cut-off point decreases, the sensitivity increases and specificity decreases. It could be observed from the result when cut-off point = 0.1 and cut-off point = 0.01:

Cell Contents			
	Count		
	Row Percent		
Total Observations in Table: 1000			
dtest$defaultn	dtest$pred.c No	Yes	Row Total
0	899	67	966
	93.064%	6.936%	96.600%
1	10	24	34
	29.412%	70.588%	3.400%
Column Total	909	91	1000

Figure 3.4.7: Actual result vs Prediction for cut-off = 0.1

```
   Cell Contents
|-------------------------|
|                   Count |
|             Row Percent |
|-------------------------|

Total Observations in Table:  1000

               | dtest$pred.c
dtest$defaultn |       No  |      Yes  | Row Total |
---------------|-----------|-----------|-----------|
             0 |      711  |      255  |      966  |
               |   73.602% |   26.398% |   96.600% |
---------------|-----------|-----------|-----------|
             1 |        1  |       33  |       34  |
               |    2.941% |   97.059% |    3.400% |
---------------|-----------|-----------|-----------|
  Column Total |      712  |      288  |     1000  |
---------------|-----------|-----------|-----------|
```

Figure 3.4.8: Actual result vs Prediction for cut-off = 0.01

It is possible to find a cut-off point such that it yields sensitivity = 100%. However, it also comes with an expense that the specificity will drop as well. Thus it is important to set priority on what are the metrics should one take in order to determine the most optimal cut-off point for this data.

3.5. Comparison of LDA, QDA, KNN, and Logistic Regression

Method	Chosen Cut-off/ K	Sensitivity	Specificity	Proportion of correct classification	Test error
LDA	0.039	0.9030	0.8592	0.861	0.139
QDA	0.038	0.8997	0.8630	0.865	0.135
KNN	15	0.3824	0.9886	0.968	0.032
Logistic Regression	0.01	0.9706	0.7360	0.744	0.256

Table 3.5.1: Comparison of performance of methods

One acknowledged that there is a trade-off between sensitivity and specificity as well as its impact to the proportion of correct classification. Moreover, one strongly believes that maximizing sensitivity with the most optimal proportion of correct classification is of the utmost importance. It is misleading if a model predicts only a few people will default on his/her credit card but in fact, there are much more

people who will default on his/her credit card. One believes that it is better to predict that there will be more default cases than the actual default cases.

By prioritizing the metrics that one should take in order to determine which is the best-fit model, one concludes that LDA is the model that fits the Default dataset best. Even though Logistic Regression yields a higher sensitivity, it has the lowest proportion of correct classification. Note that there are 34 default cases and 1000 total cases in the test data. There is roughly 7% drop in sensitivity when one utilizes LDA as compared to Logistic regression. This 7% of 34 people is approximately 2 people. However, the proportion of correct classification increases by 11.7% when one utilizes LDA as compared to Logistic Regression. This 11.7% of 1000 people is approximately 117 people. Thus selecting Logistic Regression over LDA does not worth that much because it only increases 7% in sensitivity but decreases 11.7% in proportion of correct classification. Since one prioritized in maximizing sensitivity with the most optimal proportion of correct classification, thus one opted LDA as the best-fit model to this Default dataset.

4. Appendix

```
## load all libraries

> library(MASS)

> library(ISLR)

> library(class)

> library(gmodels)

> library(ROCR)


## PART 1

> mba <- read.csv("Admission.csv")

> head(mba)


> set.seed(0820)

> dim(mba)


> std_gpa <- scale(mba$GPA)

> std_gmat <- scale(mba$GMAT)

> std_mba <- data.frame(mba$Decision, std_gpa, std_gmat)


> n = length(std_mba$mba.Decision)
```

```
> nt = 70
> train <- sample(1:n, nt)
> test <- std_mba[-train,]
> training <- std_mba[train,]

## LDA
> ldafit <- lda(mba.Decision ~ std_gpa + std_gmat, data = std_mba, subset = train)
> ldafit

> ldatestpred <- predict(ldafit, test)
> ldatrainpred <- predict(ldafit, training)

> postest <- as.data.frame(round(ldatestpred$posterior), digits = 4)
> postrain <- as.data.frame(round(ldatrainpred$posterior, digits = 4))

> ldatestclass <- ldatestpred$class
> ldatrainclass <- ldatrainpred$class
> table(ldatestclass, test$mba.Decision)
> table(ldatrainclass, training$mba.Decision)

> mean(ldatestclass == test$mba.Decision)

> qdafit <- qda(mba.Decision ~ std_gpa + std_gmat, data = std_mba, subset = train)
> qdafit

> qdatestpred <- predict(qdafit, test)
> qdatrainpred <- predict(qdafit, training)
> qpostest <- as.data.frame(round(qdatestpred$posterior, digits = 4))
> qpostrain <- as.data.frame(round(qdatrainpred$posterior, digits = 4))
```

```
> qdatestclass <- qdatestpred$class

> qdatrainclass <- qdatrainpred$class

> table(qdatestclass, test$mba.Decision)

> table(qdatrainclass, training$mba.Decision)


> mean(qdatestclass == test$mba.Decision)


## PART 2

> set.seed(0820)

> Default$defaultn <- as.numeric(Default$default)-1

> Default$studentn <- as.numeric(Default$student)-1

> Default$balancer <- Default$balance/1000

> Default$incomer <- Default$income/1000


> n = length(Default$defaultn)

> nt = 9000

> dtrain <- sample(1:n, nt)

> dtest <- Default[-dtrain,]

> dtraining <- Default[dtrain,]


## LDA

> dldafit <- lda(defaultn ~ studentn + balancer, data = Default, subset = dtrain)

> dldafit


> dldatestpred <- predict(dldafit, dtest)

> dldatrainpred <- predict(dldafit, dtraining)


> dprobtest <- as.data.frame(round(dldatestpred$posterior, digits = 4))
```

```
> dprobtrain <- as.data.frame(round(dldatrainpred$posterior, digits = 4))

> dprobtestclass <- dldatestpred$class
> dprobtrainclass <-dldatrainpred$class
> table(dprobtestclass, dtest$defaultn)
> mean(dprobtestclass == dtest$defaultn)

> sensitivity=dim(1000)
> specificity=dim(1000)
> total=dim(1000)
> cutoff=dim(1000)

> for (k in 1:1000) {
+ p <- k/1000
+ pred <- dprobtrain[,2]
+ predg <- as.numeric(pred >= p)
+ sens <- sum(dtraining$defaultn*predg)/sum(dtraining$defaultn)
+ spec <- 1-sum((1-dtraining$defaultn)*predg)/sum(1-dtraining$defaultn)
+ sensitivity[k]=sens
+ specificity[k]=spec
+ total[k]=sens+spec
+ cutoff[k]=p
+ }
> evallda <- data.frame(cutoff,total,sensitivity,specificity)
> evallda[which(evallda$total == max(evallda$total)),]

## QDA
> dqdafit <- qda(defaultn ~ studentn + balancer, data = Default, subset = dtrain)
> dqdafit
```

```
> dqdatestpred <- predict(dqdafit, dtest)
> dqdatrainpred <- predict(dqdafit, dtraining)

> dqprobtest <- as.data.frame(round(dqdatestpred$posterior, digits = 4))
> dqprobtrain <- as.data.frame(round(dqdatrainpred$posterior, digits = 4))

> dqprobtestclass <- dqdatestpred$class
> dqprobtrainclass <-dqdatrainpred$class
> table(dqprobtestclass, dtest$defaultn)
> mean(dqprobtestclass == dtest$defaultn)

> sensitivity=dim(1000)
> specificity=dim(1000)
> total=dim(1000)
> cutoff=dim(1000)

> for (k in 1:1000) {
+ p <- k/1000
+ pred <- dqprobtrain[,2]
+ predg <- as.numeric(pred >= p)
+ sens <- sum(dtraining$defaultn*predg)/sum(dtraining$defaultn)
+ spec <- 1-sum((1-dtraining$defaultn)*predg)/sum(1-dtraining$defaultn)
+ sensitivity[k]=sens
+ specificity[k]=spec
+ total[k]=sens+spec
+ cutoff[k]=p
+ }
> evallda <- data.frame(cutoff,total,sensitivity,specificity)
```

```
> evallda[which(evallda$total == max(evallda$total)),]
```

KNN

```
> attach(Default)
> nearest1 <- knn(train=predictors[train,],test=predictors[-train,],cl=defaultn[train],k=1)
> nearest3 <- knn(train=predictors[train,],test=predictors[-train,],cl=defaultn[train],k=3)
> nearest5 <- knn(train=predictors[train,],test=predictors[-train,],cl=defaultn[train],k=5)
> nearest7 <- knn(train=predictors[train,],test=predictors[-train,],cl=defaultn[train],k=7)
> nearest9 <- knn(train=predictors[train,],test=predictors[-train,],cl=defaultn[train],k=9)
> nearest11 <- knn(train=predictors[train,],test=predictors[-train,],cl=defaultn[train],k=11)
> nearest13 <- knn(train=predictors[train,],test=predictors[-train,],cl=defaultn[train],k=13)
> nearest15 <- knn(train=predictors[train,],test=predictors[-train,],cl=defaultn[train],k=15)

> results <- data.frame(Default[-train],nearest1,nearest3,nearest5,nearest7, nearest9,
nearest11,nearest13,nearest15)
> results

> pcorrn1 <- sum(defaultn[-train] == nearest1) / (n-nt)
> pcorrn3 <- sum(defaultn[-train] == nearest3) / (n-nt)
> pcorrn5 <- sum(defaultn[-train] == nearest5) / (n-nt)
> pcorrn7 <- sum(defaultn[-train] == nearest7) / (n-nt)
> pcorrn9 <- sum(defaultn[-train] == nearest9) / (n-nt)
> pcorrn11 <- sum(defaultn[-train] == nearest11) / (n-nt)
> pcorrn13 <- sum(defaultn[-train] == nearest13) / (n-nt)
> pcorrn15 <- sum(defaultn[-train] == nearest15) / (n-nt)
> correct <- data.frame(pcorrn1, pcorrn3, pcorrn5, pcorrn7, pcorrn9, pcorrn11, pcorrn13, pcorrn15)
> correct

> predictors <- data.frame(studentn, balancer)
```

```
> pcorr = dim(15)
> for (k in 1:15) {
+ pred = knn.cv(predictors[dtrain,],defaultn[dtrain],k)
+ pcorr[k] = sum(defaultn[dtrain] == pred) / n
+ }
> plot(pcorr, type = "b")

> table(nearest13, defaultn[-train])
> table(nearest15, defaultn[-train])

## LOGISTIC REGRESSION
> logm <- glm(defaultn ~ studentn + balancer, data = Default, subset = dtrain, family = binomial)
> summary(logm)
> drop1(logm, test = "LRT")

> dtest$pred <- predict(logm, dtest, type = "response")
> dtest$pred.c <- ifelse(dtest$pred >= 0.5, c("Yes"), c("No"))
> CrossTable(dtest$defaultn,dtest$pred.c,expected=FALSE, prop.chisq=FALSE, prop.t=FALSE,
prop.c=FALSE, chisq=FALSE, format="SPSS")

> Default$pred <- predict(logm, dtest, type = "response")

> pred <- prediction(Default$pred,Default$defaultn)
> perf <- performance(pred,"tpr","fpr")
> plot(perf, col="red", lty="solid", lwd=2, xaxs="i", yaxs="i", main="ROC- curve")
> abline(a=0,b=1, lwd=2)
```