PREDICTIVE ACCURACY COMPARISION OF DATA MINNING TECHNIQUES FOR DEFAULT OF CREDIT CARD CLIENTS

Group No.: Group 23

Student Names: Devarsh Shah and Gaurav Handa

Executive Summary: This report has been conducted to compare predictive accuracy of the most well-known data-mining techniques used for classification problems namely; K- Nearest Neighbor, Linear Discriminant Analysis, Naive Bayes, Support Vector Machine, Neural networks, Logistic regression and Random Forest. This report discusses key concepts of each technique, its advantages and disadvantages. Further, Practical application of these techniques is seen to generate and evaluate the performance of these techniques.

I. Background and Introduction

Predicting the default of credit card is crucial for any financial institution to financially stable. Systems which can predict the default of customer based on his current transactions and his ability to regularly pay the bills generated, can flourish well. The major purpose of risk prediction is to use financial information, such as business financial statement, customer transaction and repayment records, etc., to predict business performance or individual customers' credit risk and to reduce the damage and uncertainty. While faulty systems can over issue cards to unqualified customers irrespective of their repayment ability overused credit card for consumption and accumulated heavy credit and cash– card debts. The crisis caused the blow to consumer finance confidence

- The problem: Data which is collected of customers is highly varied and complex and investigates major aspects of their life which can be useful to understand predict their activity of repayment. This complex data needs to be accurate and sensible for system to make better predictions.

- The goal of your study: Goal of this study is taking numerous previous activities of customers and their repayment ability into account to accurately predict if each customer will face default of credit card.

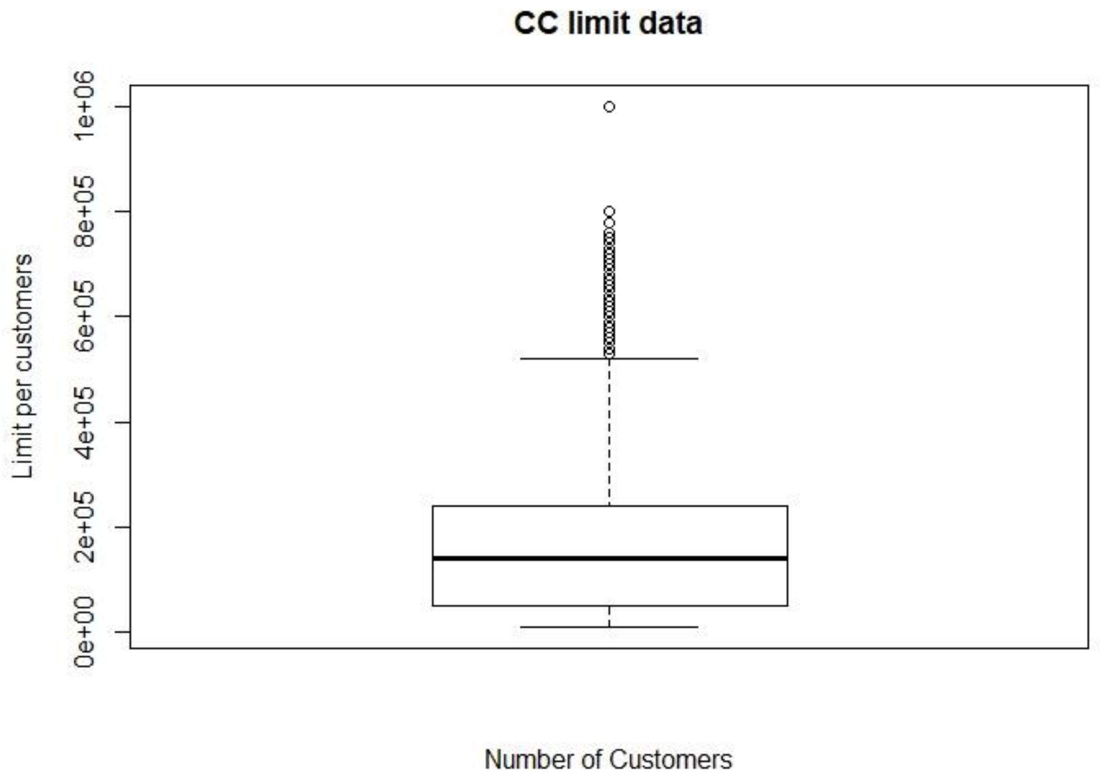
- The possible solution: Possible solution is use machine learning and advanced data-mining techniques to accurately reach the goal and compare which technique is better for the given problem.

II. Data Exploration and Visualization

One of the earliest stages of engaging with a dataset is exploring it. Exploration is aimed at understanding the global landscape of the data and detecting unusual values. Exploration is used for data cleaning and manipulation as well as for visual discovery and "hypothesis generation." This includes looking at each variable separately as well as looking at relationships among variables. The purpose is to discover patterns and exceptions.

1.Boxplot

For some distributions/datasets, you will find that you need more information than the measures of central tendency (median, mean, and mode). You need to have information on the variability or dispersion of the data. A boxplot is a graph that gives you a good indication of how the values in the data are spread out. Although boxplots may seem primitive in comparison to a histogram or density plot, they have the advantage of taking up less space, which is useful when comparing distributions between many groups or datasets. Boxplots are a standardized way of displaying the distribution of data based on a five-number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”).

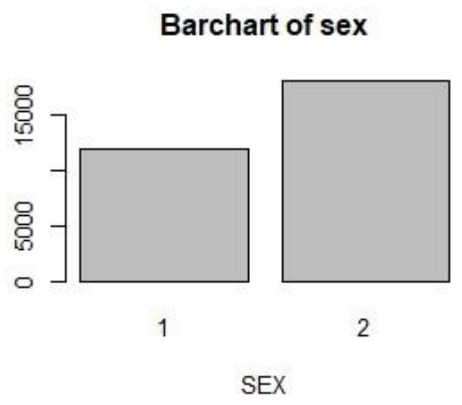


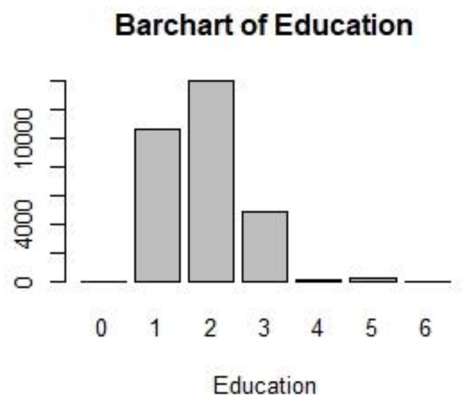
2. Histogram

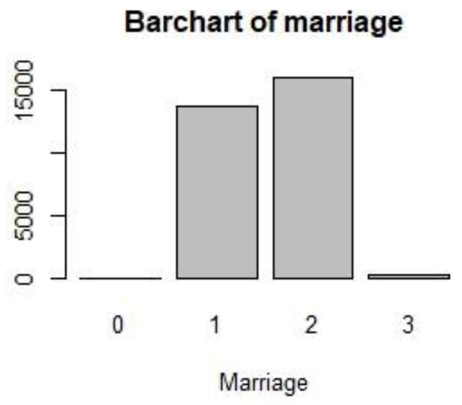
A histogram is a plot that lets you discover, and show, the underlying frequency distribution (shape) of a set of continuous data. This allows the inspection of the data for its underlying distribution (e.g., normal distribution), outliers, skewness, etc.

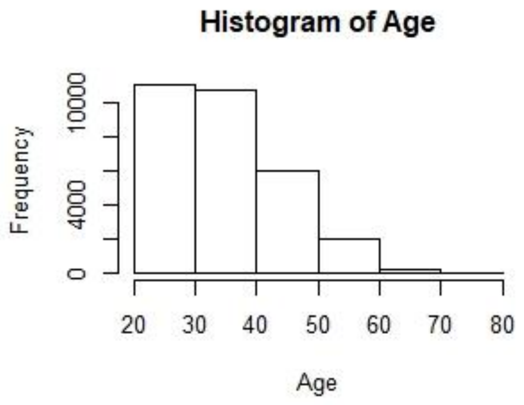
3. Bar chart

A bar chart is a way of summarizing a set of categorical data (continuous data can be made categorical by auto-binning). The bar chart displays data using several bars, each representing a category. The height of each bar is proportional to a specific aggregation (for example the sum of the values in the category it represents).









III. Data Preparation and Preprocessing

Our study took payment data in October, 2005, from an important bank (a cash and credit card issuer) in Taiwan and the targets were credit card holders of the bank. Here 30,000 observations have been recorded of customers. This research employed a binary variable – default payment (Yes = 1, No = 0), as the response variable. 23 Predictors are used here:

X1: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.
X2: Gender (1 = male; 2 = female).
X3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).
X4: Marital status (1 = married; 2 = single; 3 = others).
X5: Age (year).
X6–X11: History of past payment. We tracked the past monthly payment records (from April to September, 2005) as follows:
X6 = the repayment status in September, 2005;
X7 = the repayment status in August, 2005;...;X11 = the repayment status in April, 2005.

The measurement scale for the repayment status is: 1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; ...; 8 = payment delay for eight months; 9 = payment delay for nine months and above.

X12–X17: Amount of bill statement (NT dollar).
X12 = amount of bill statement in September, 2005; ,
X13 = amount of bill statement in August, 2005;...;X17 = amount of bill statement in April, 2005.
X18–X23: Amount of previous payment (NT dollar).
X18 = amount paid in September, 2005;
X19 = amount paid in August, 2005;...;X23 = amount paid in April, 2005.

Normalization

$$z = \frac{x \ \text{-} \min(x)}{\max(x) \text{ - } \min(x)}$$

Normalization is a good technique to use when you do not know the distribution of your data or when you know the distribution is not Gaussian (a bell curve). Normalization is useful when data has varying scales and the algorithm used does not make assumptions about the distribution of your data, such as k-nearest neighbors and artificial neural networks.

Standardization

Standardization:

$$z = \frac{x-\mu}{\sigma}$$

with mean:

$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)$$

and standard deviation:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Standardization assumes that your data has a Gaussian (bell curve) distribution. This does not strictly have to be true, but the technique is more effective if your attribute distribution is Gaussian. Standardization is useful when your data has varying scales and the algorithm you are using does make assumptions about your data having a Gaussian distribution, such as linear regression, logistic regression, and linear discriminant analysis.

IV. Data Mining Techniques and Implementation

K-nearest neighbor classifiers (KNN)

K-nearest neighbor (KNN) classifiers are based on learning by analogy. When given an unknown sample, a KNN classifier searches the pattern space for the KNN that are closest to the unknown sample. Closeness is defined in terms of distance. The unknown sample is assigned the most common class among its KNN. The major advantage of this approach is that it is not required to establish predictive model before classification. The disadvantages are that KNN does not produce a simple classification probability formula and its predictive accuracy is highly affected by the measure of distance and the cardinality k of the neighborhood.

Logistic regression (LR)

Logistic regression can be considered a special case of linear regression models. However, the binary response variable violates normality assumptions of general regression models. A logistic regression model specifies that an appropriate function of the fitted probability of the event is a linear function of the observed values of the available explanatory variables. The major advantage of this approach is that it can produce a simple probabilistic formula of classification. The weaknesses are that LR cannot properly deal with the problems of non-linear and interactive effects of explanatory variables.

Discriminant analysis (DA)

Discriminant analysis, also known as Fisher's rule, is another technique applied to the binary result of response variable. DA is an alternative to logistic regression and is based on the assumptions that, for each given class of response variable, the explanatory variables are distributed as a multivariate normal distribution with a common variance–covariance matrix. The objective of Fisher's rule is to maximize the distance between different groups and to minimize the distance within each group. The pros and cons of DA are similar to those of LR.

Naive Bayesian classifier (NB)

The naive Bayesian classifier is based on Bayes theory and assumes that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence. Bayesian classifiers are useful in that they provide a theoretical justification for other classifiers that do not explicitly use Bayes theorem. The major weakness of NB is that the predictive accuracy is highly correlated with the assumption of class conditional independence. This assumption simplifies computation. In practice, however, dependences can exist between variables.

Artificial neural networks (ANNs)

Artificial neural networks use non-linear mathematical equations to successively develop meaningful relationships between input and output variables through a learning process. We applied back propagation networks to classify data. A back propagation neural network uses a feed-forward topology and supervised learning. The structure of back propagation networks is typically composed of an input layer, one or more hidden layers, and an output layer, each consisting of several neurons. ANNs can easily handle the non-linear and interactive effects of explanatory variables. The major drawback of ANNs is – they cannot result in a simple probabilistic formula of classification.

Classification trees (CTs)

In a classification tree structure, each internal node denotes a test on an attribute, each branch represents an outcome of the test, and leaf nodes represent classes. The top-most node in a tree is the root node. CTs are applied when the response variable is qualitative or quantitative discrete. Classification trees perform a classification of the observations based on all explanatory variables and supervised by the presence of the response variable. The segmentation process is typically carried out using only one explanatory variable at a time. CTs are based on minimizing impurity, which refers to a measure of variability of the response values of the observations. CTs can result in simple classification rules and can handle the nonlinear and interactive effects of explanatory variables. But their sequential nature and algorithmic complexity can make them depends on the observed data, and even a small change might alter the structure of the tree. It is difficult to take a tree structure designed for one context and generalize it for other contexts.

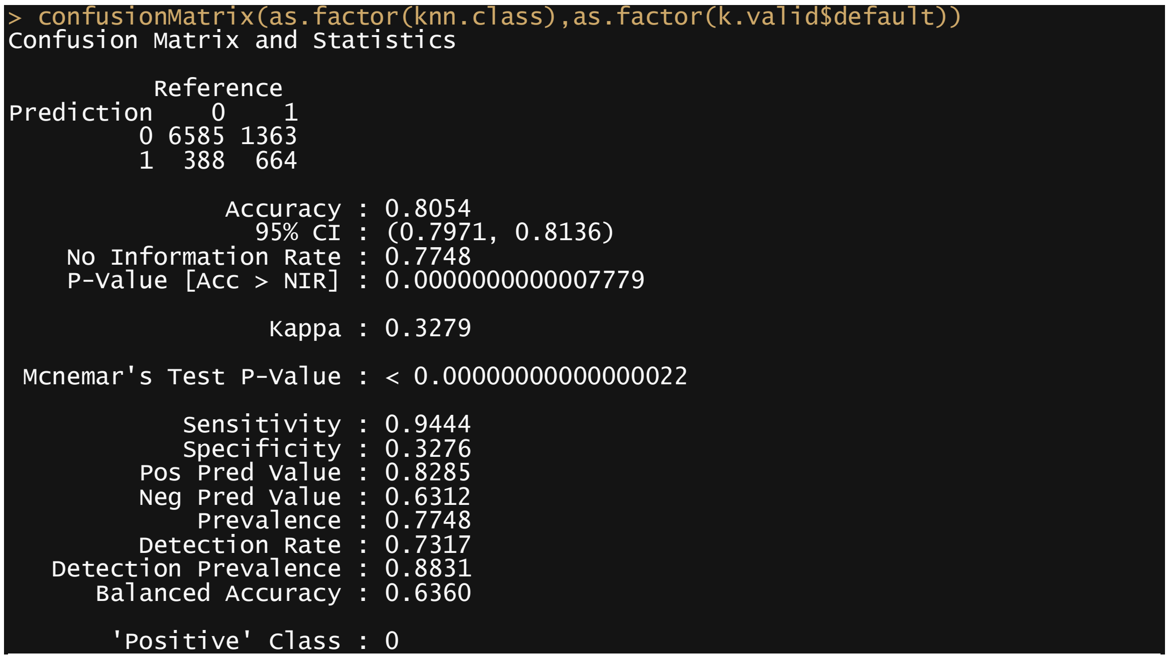
Support Vector Machine (SVM)

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well. Support Vectors are simply the co-ordinates of individual observation. The SVM classifier is a frontier which best segregates the two classes (hyper-plane/ line).

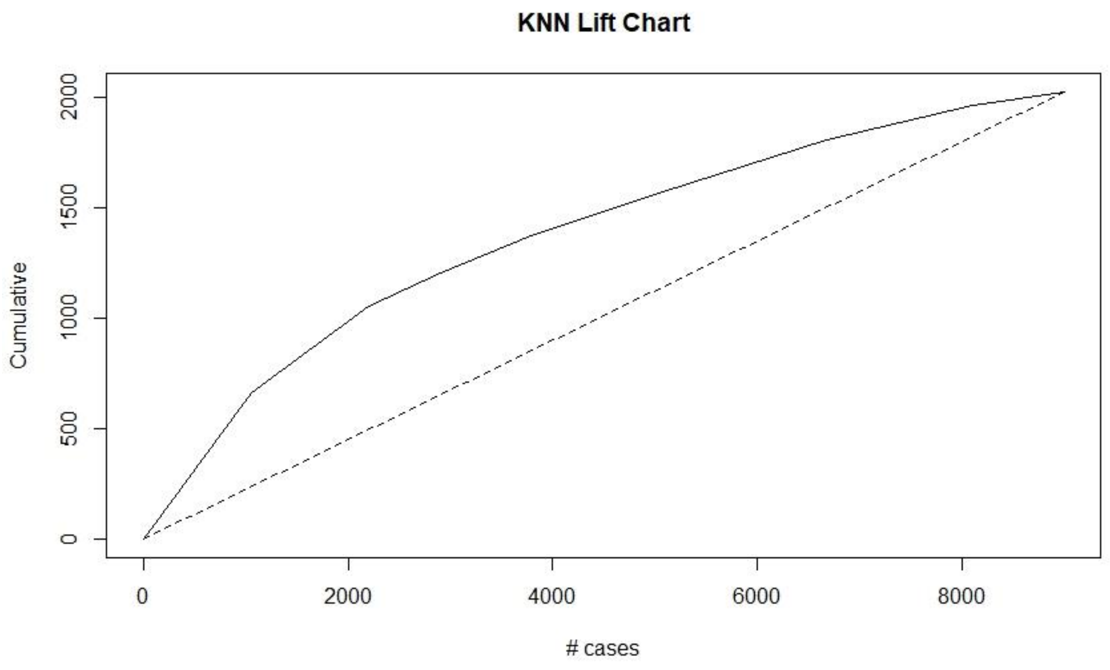
V. Performance Evaluation

Here we use Confusion Matrix and Lift chart to compare the performance of each algorithm.

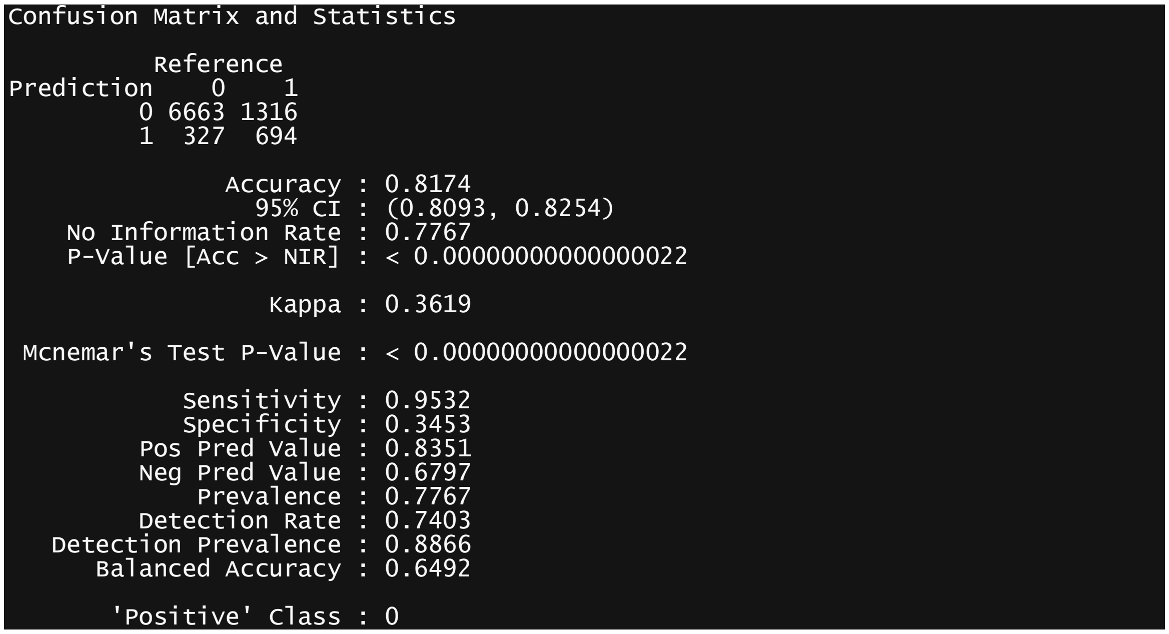
K-nearest neighbor classifiers (KNN)



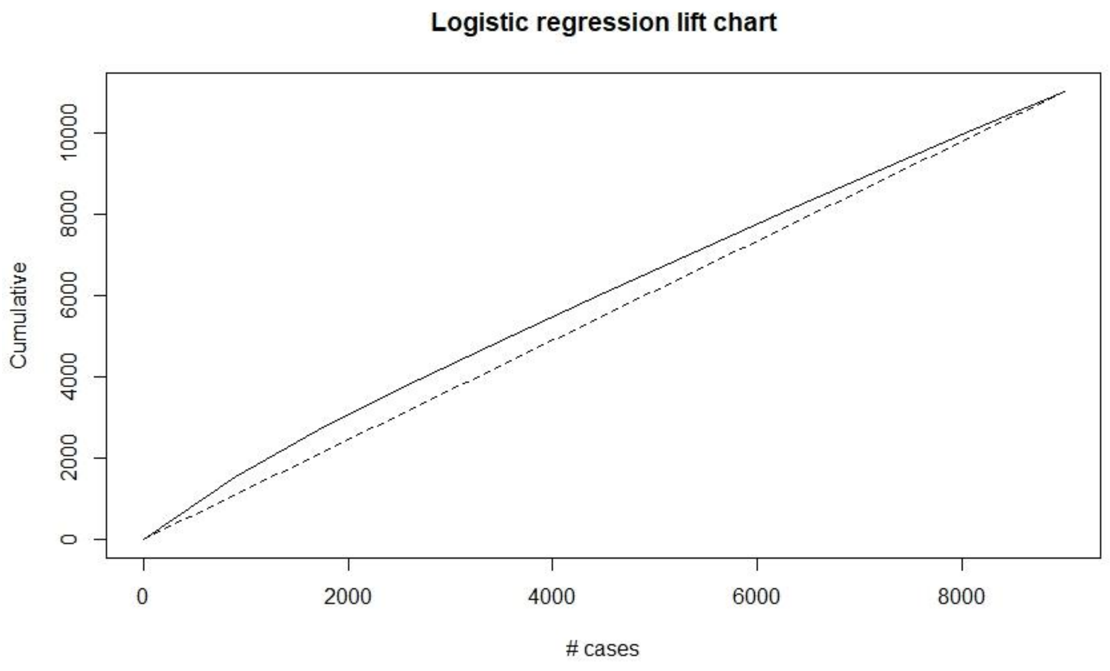
```
> confusionMatrix(as.factor(knn.class),as.factor(k.valid$default))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6585 1363
         1  388  664

               Accuracy : 0.8054
                 95% CI : (0.7971, 0.8136)
    No Information Rate : 0.7748
    P-Value [Acc > NIR] : 0.0000000000007779

                  Kappa : 0.3279

 Mcnemar's Test P-Value : < 0.00000000000000022

            Sensitivity : 0.9444
            Specificity : 0.3276
         Pos Pred Value : 0.8285
         Neg Pred Value : 0.6312
             Prevalence : 0.7748
         Detection Rate : 0.7317
   Detection Prevalence : 0.8831
      Balanced Accuracy : 0.6360

       'Positive' Class : 0
```



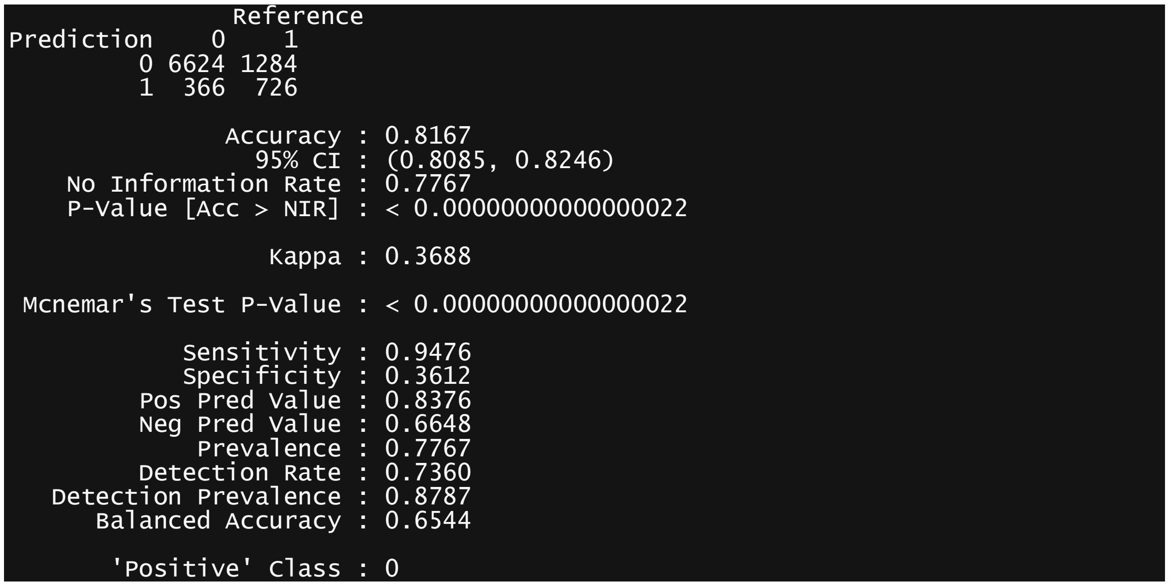
Logistic regression (LR)



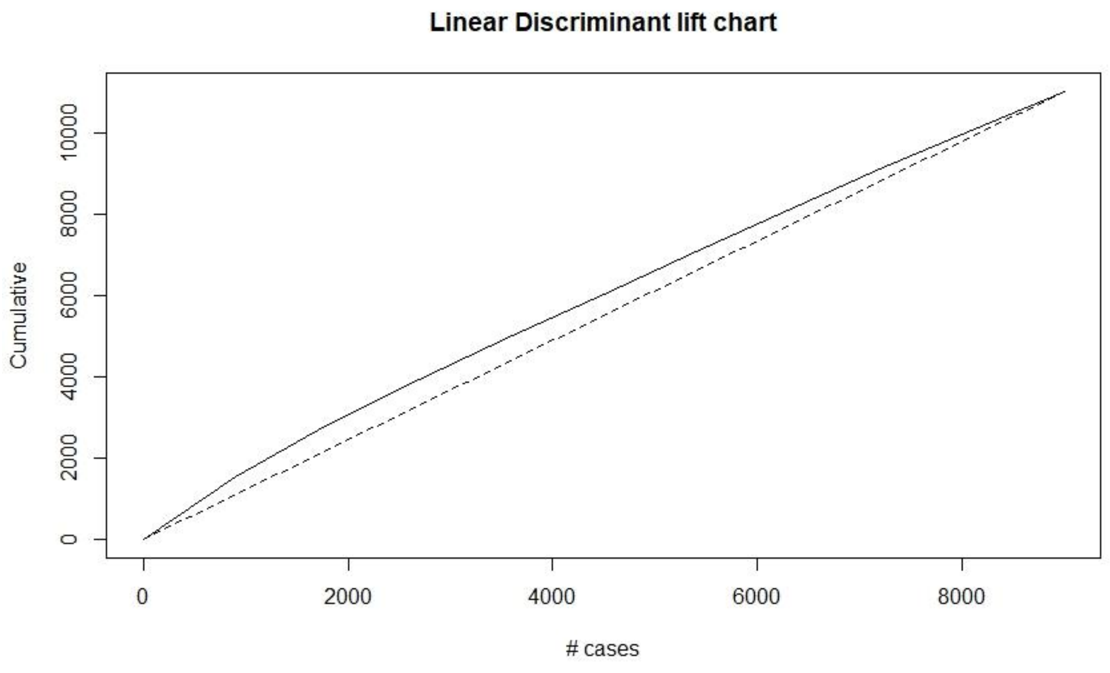
```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6663 1316
         1  327  694

               Accuracy : 0.8174
                 95% CI : (0.8093, 0.8254)
    No Information Rate : 0.7767
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.3619

 Mcnemar's Test P-Value : < 0.00000000000000022

            Sensitivity : 0.9532
            Specificity : 0.3453
         Pos Pred Value : 0.8351
         Neg Pred Value : 0.6797
             Prevalence : 0.7767
         Detection Rate : 0.7403
   Detection Prevalence : 0.8866
      Balanced Accuracy : 0.6492

       'Positive' Class : 0
```



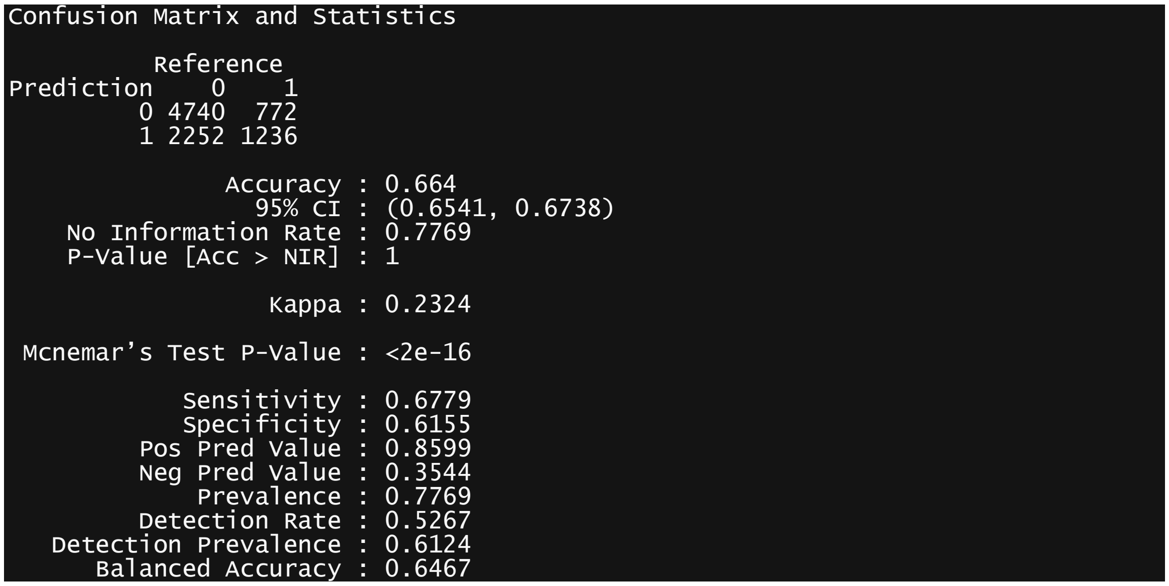
Discriminant analysis (DA)



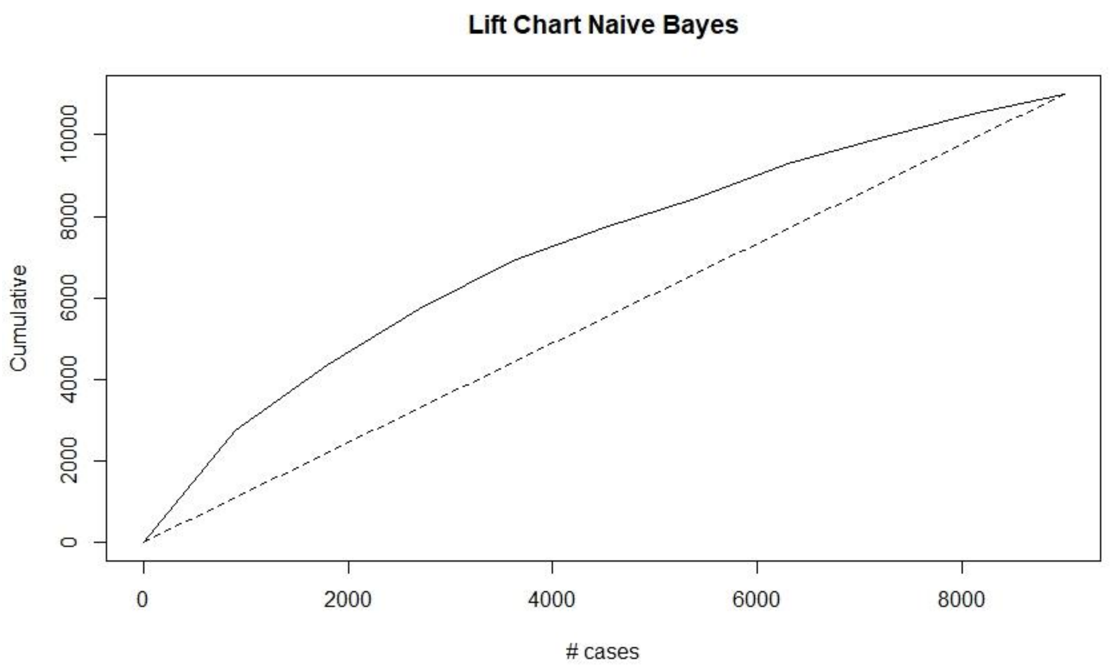
```
               Reference
Prediction    0    1
         0 6624 1284
         1  366  726

               Accuracy : 0.8167
                 95% CI : (0.8085, 0.8246)
    No Information Rate : 0.7767
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.3688

 Mcnemar's Test P-Value : < 0.00000000000000022

            Sensitivity : 0.9476
            Specificity : 0.3612
         Pos Pred Value : 0.8376
         Neg Pred Value : 0.6648
             Prevalence : 0.7767
         Detection Rate : 0.7360
   Detection Prevalence : 0.8787
      Balanced Accuracy : 0.6544

       'Positive' Class : 0
```



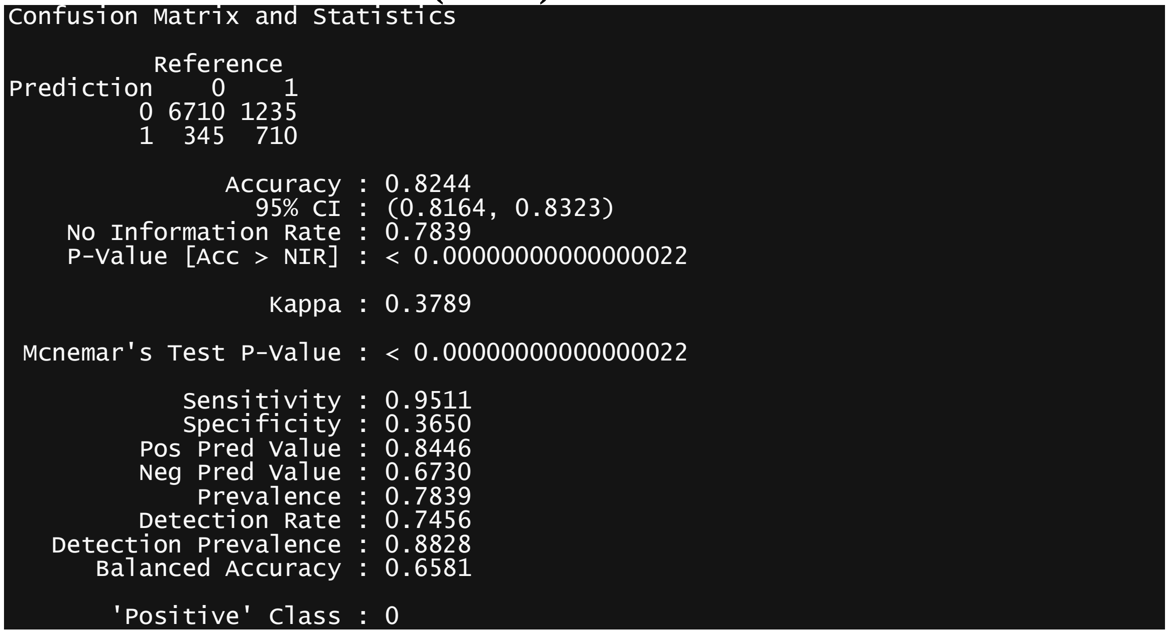
Naïve Bayesian classifier (NB)



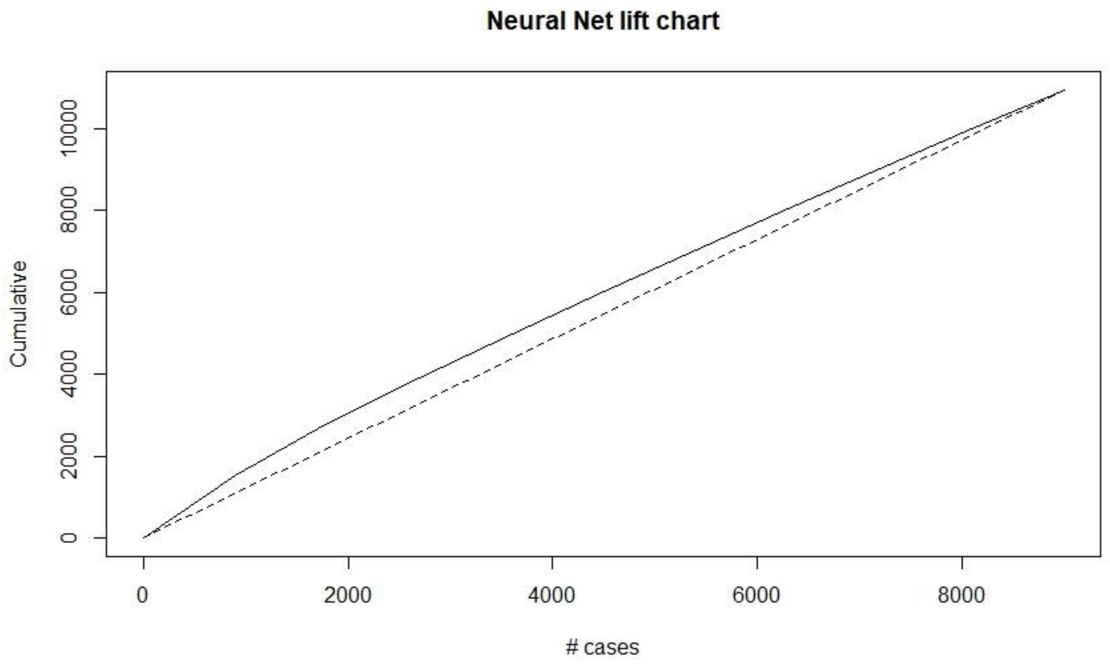
```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 4740  772
         1 2252 1236

               Accuracy : 0.664
                 95% CI : (0.6541, 0.6738)
    No Information Rate : 0.7769
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2324

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.6779
            Specificity : 0.6155
         Pos Pred Value : 0.8599
         Neg Pred Value : 0.3544
             Prevalence : 0.7769
         Detection Rate : 0.5267
   Detection Prevalence : 0.6124
      Balanced Accuracy : 0.6467
```



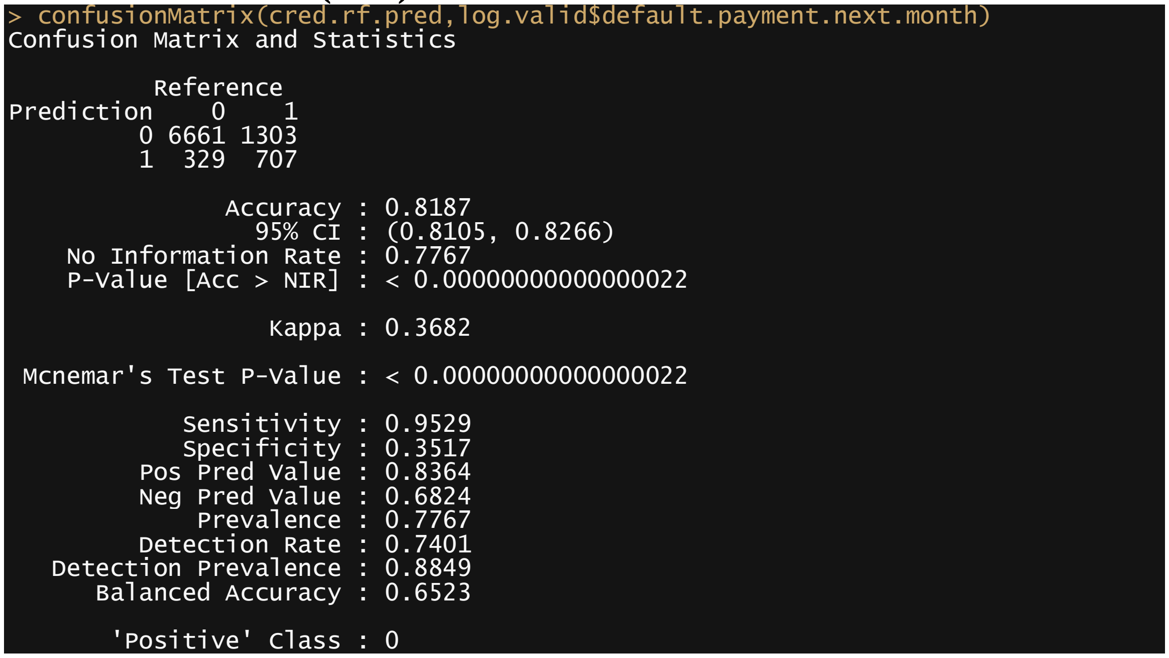
Artificial neural networks (ANNs)



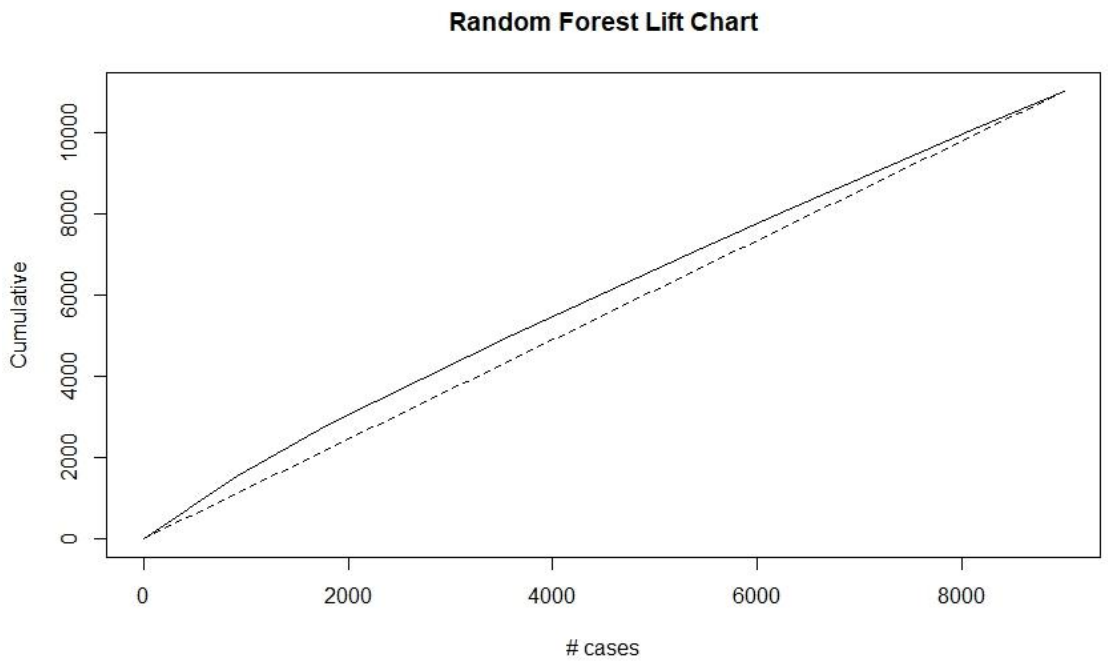
```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6710 1235
         1  345  710

               Accuracy : 0.8244
                 95% CI : (0.8164, 0.8323)
    No Information Rate : 0.7839
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.3789

 Mcnemar's Test P-Value : < 0.00000000000000022

            Sensitivity : 0.9511
            Specificity : 0.3650
         Pos Pred Value : 0.8446
         Neg Pred Value : 0.6730
             Prevalence : 0.7839
         Detection Rate : 0.7456
   Detection Prevalence : 0.8828
      Balanced Accuracy : 0.6581

       'Positive' Class : 0
```



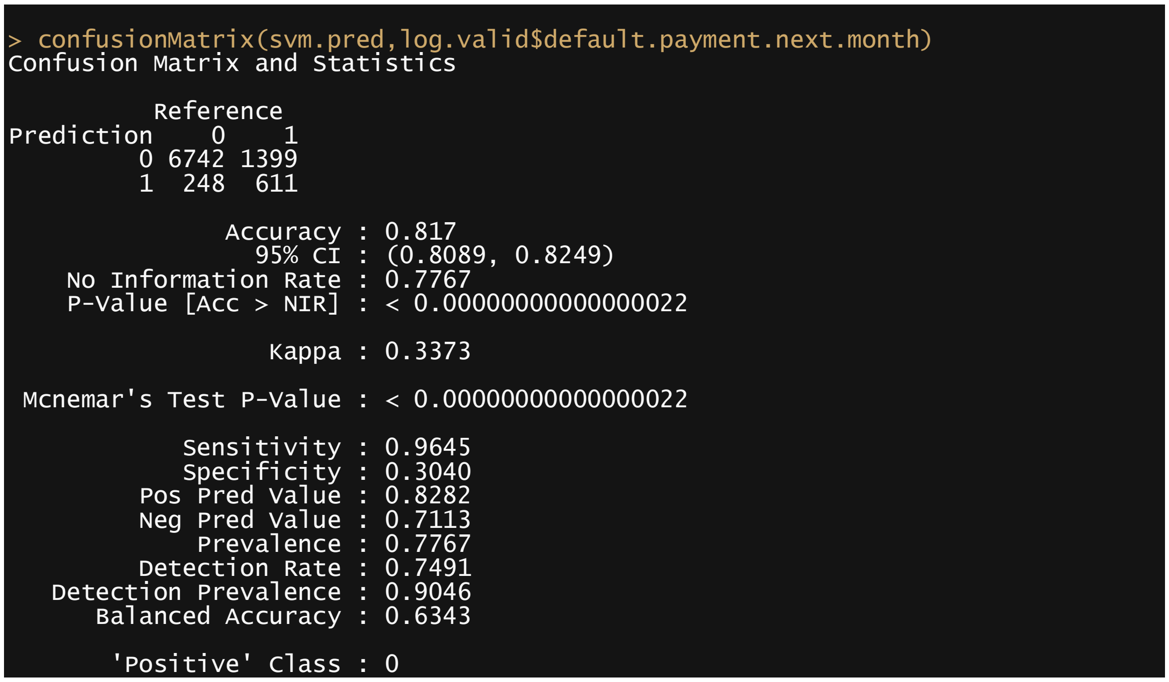
Classification trees (CTs)



```
> confusionMatrix(cred.rf.pred,log.valid$default.payment.next.month)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6661 1303
         1  329  707

               Accuracy : 0.8187
                 95% CI : (0.8105, 0.8266)
    No Information Rate : 0.7767
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.3682

 Mcnemar's Test P-Value : < 0.00000000000000022

            Sensitivity : 0.9529
            Specificity : 0.3517
         Pos Pred Value : 0.8364
         Neg Pred Value : 0.6824
             Prevalence : 0.7767
         Detection Rate : 0.7401
   Detection Prevalence : 0.8849
      Balanced Accuracy : 0.6523

       'Positive' Class : 0
```



Support Vector Machine (SVM)



```
> confusionMatrix(svm.pred,log.valid$default.payment.next.month)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 6742 1399
         1  248  611
                                              
               Accuracy : 0.817               
                 95% CI : (0.8089, 0.8249)    
    No Information Rate : 0.7767              
    P-Value [Acc > NIR] : < 0.00000000000000022
                                              
                  Kappa : 0.3373              
                                              
 Mcnemar's Test P-Value : < 0.00000000000000022
                                              
            Sensitivity : 0.9645              
            Specificity : 0.3040              
         Pos Pred Value : 0.8282              
         Neg Pred Value : 0.7113              
             Prevalence : 0.7767              
         Detection Rate : 0.7491              
   Detection Prevalence : 0.9046              
      Balanced Accuracy : 0.6343              
                                              
       'Positive' Class : 0                   
```

Here we couldn’t generate the lift chart for SVM as model for SVM only generates the class for prediction but not the probabilities.

VI. Discussion and Recommendation

From the accuracy reports from confusion matrix it can be seen that ANN performs the best. Neural networks can linear and non-linear programming problems. No prior knowledge of process generating the data is required to be applied and they perform the best for this kind of problem.

Naïve Bayes performs the worst. Binning the data and assigning discrete classes to the bins is sub-optimal since it throws away information. Assuming each feature is normally distributed is workable but could impact performance if features are not normally distributed. On the other hand, with enough training data in each class, you could estimate the likelihood densities directly, permitting accurate likelihood calculations for new data.

Performance is sensitive to skewed data — that is, when the training data is not representative of the class distributions in the overall population. In this case, the prior estimates will be incorrect.

VII. Summary

This paper examines the six major classification techniques in data mining and compares the performance of classification and predictive accuracy among them. From this case study we can see that ANN perform well for default of credit cards. In the classification accuracy among the six data mining techniques, the results show that there are little differences in error rates among the six methods. The predictive default probability produced by ANN is the only one that could be used to represent real probability of default. Therefore, artificial neural networks should be employed to score clients instead of other data mining techniques, such as logistic regression.

Appendix: R Code for use case study

```
 cred<- read.csv("C:/Users/Gaurav/Downloads/default of credit card clients
(1).csv",header = T,skip=1)
View(cred)

summary(cred)
cred<- cred[-1,]
str(cred)
cred$SEX<- factor(cred$SEX)
cred$MARRIAGE<- factor(cred$MARRIAGE)
cred$PAY_0<- factor(cred$PAY_0)
cred$PAY_2<- factor(cred$PAY_2)
cred$PAY_3<- factor(cred$PAY_3)
cred$PAY_4<- factor(cred$PAY_4)
cred$PAY_5<- factor(cred$PAY_5)
cred$PAY_6<- factor(cred$PAY_6)
cred$EDUCATION<- factor(cred$EDUCATION)
cred$default.payment.next.month<- factor(cred$default.payment.next.month)
nv.cred<- cred
nv.cred$AGE<- factor(round(nv.cred$AGE/100))
nv.cred$LIMIT_BAL<- factor(round(nv.cred$LIMIT_BAL/100))
nv.cred$BILL_AMT1<- factor(round(nv.cred$BILL_AMT1/100))
nv.cred$BILL_AMT2<- factor(round(nv.cred$BILL_AMT2/100))
nv.cred$BILL_AMT3<- factor(round(nv.cred$BILL_AMT3/100))
nv.cred$BILL_AMT4<- factor(round(nv.cred$BILL_AMT4/100))
nv.cred$BILL_AMT5<- factor(round(nv.cred$BILL_AMT5/100))
nv.cred$BILL_AMT6<- factor(round(nv.cred$BILL_AMT6/100))
nv.cred$PAY_AMT1<- factor(round(nv.cred$PAY_AMT1/100))
nv.cred$PAY_AMT2<- factor(round(nv.cred$PAY_AMT2/100))
nv.cred$PAY_AMT3<- factor(round(nv.cred$PAY_AMT3/100))
nv.cred$PAY_AMT4<- factor(round(nv.cred$PAY_AMT4/100))
nv.cred$PAY_AMT5<- factor(round(nv.cred$PAY_AMT5/100))
nv.cred$PAY_AMT6<- factor(round(nv.cred$PAY_AMT6/100))
View(nv.cred)
nv.cred<- nv.cred[,-1]
str(nv.cred)
library(e1071)
set.seed(10)
nv.train.index <- sample(c(1:dim(nv.cred)[1]), dim(nv.cred)[1]*0.7)
nv.train<- nv.cred[nv.train.index, ]
nv.valid <- nv.cred[-nv.train.index, ]
naive.cred<- naiveBayes(default.payment.next.month ~ .,data= nv.train)
View(nv.train)
nvpred.prob <- predict(naive.cred, newdata = nv.valid, type = "raw")

nvpred.class <- predict(naive.cred, newdata = nv.valid)
```

```
View(nvpred.prob)
nvdf <- data.frame(actual = nv.valid$default.payment.next.month, predicted =
nvpred.class, nvpred.prob)
View(nvdf)

confusionMatrix(nvpred.class, nv.valid$default.payment.next.month)
library(gains)
nvgain <- gains(as.integer(as.vector(nvdf$actual)),nvdf[,4],groups=10)
plot(c(0,as.integer(nvgain$cume.pct.of.total*sum(as.integer(nvdf$actual))))~c(0,nvgain$
cume.obs),
    xlab="# cases", ylab="Cumulative", main="Lift Chart Naive Bayes", type="l")
lines(c(0,sum(as.integer(nvdf$actual)))~c(0, dim(nvdf)[1]), lty=2)

logit.cred<- cred
set.seed(45)
logtrindex<- sample(c(1:dim(logit.cred)[1]), dim(logit.cred)[1]*0.7)
log.train<- logit.cred[logtrindex, ]
log.valid <- logit.cred[-logtrindex, ]
log.train<- log.train[,-1]
log.valid<- log.valid[,-1]
logit.reg <- glm(default.payment.next.month ~ ., data = log.train, family = "binomial")
options(scipen=999)
logit.reg.pred <- predict(logit.reg, log.valid[, -24], type = "response")
loggain <- gains(as.integer(log.valid$default.payment.next.month), logit.reg.pred,
groups=10)
plot(c(0,loggain$cume.pct.of.total*sum(as.integer(log.valid$default.payment.next.month)
))~c(0,loggain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="Logistic regression lift chart", type="l")
lines(c(0,sum(as.integer(log.valid$default.payment.next.month)))~c(0,
dim(log.valid)[1]), lty=2)
logdf<- data.frame(logit.reg.pred,log.valid$default.payment.next.month)
View(logdf)
logclass<-ifelse(logit.reg.pred<0.5,0,1)
logclass<-as.factor(logclass)
confusionMatrix(logclass,log.valid$default.payment.next.month)

library(MASS)
lda.cred<- lda(default.payment.next.month~., log.train)
lda.pred<- predict(lda.cred, log.valid[,-24])
ldagain <- gains(as.integer(log.valid$default.payment.next.month),
lda.pred$posterior[,2], groups=10)
plot(c(0,ldagain$cume.pct.of.total*sum(as.integer(log.valid$default.payment.next.month)
))~c(0,ldagain$cume.obs),
    xlab="# cases", ylab="Cumulative", main="Linear Discriminant lift chart", type="l")
lines(c(0,sum(as.integer(log.valid$default.payment.next.month)))~c(0,
dim(log.valid)[1]), lty=2)
```

```
confusionMatrix(lda.pred$class,log.valid$default.payment.next.month)

library(randomForest)
cred.rf<- randomForest(default.payment.next.month~., log.train, ntree = 500, mtry = 4,
nodesize = 5, importance = TRUE)
#
cred.rf.pred <- predict(cred.rf, log.valid[,-24])
View(cred.rf.pred)
confusionMatrix(cred.rf.pred,log.valid$default.payment.next.month)
cred.rf.pred1 <- predict(cred.rf, log.valid[,-24],type = "prob")
rfgain <- gains(as.integer(log.valid$default.payment.next.month), cred.rf.pred1[,2],
groups=10)
plot(c(0,rfgain$cume.pct.of.total*sum(as.integer(log.valid$default.payment.next.month))
)~c(0,rfgain$cume.obs),
     xlab="# cases", ylab="Cumulative", main="Random Forest Lift Chart", type="l")
lines(c(0,sum(as.integer(log.valid$default.payment.next.month)))~c(0,
dim(log.valid)[1]), lty=2)



svm.cred<- svm(default.payment.next.month~.,data = log.train)
svm.pred<- predict(svm.cred,log.valid[,-24])
svm.pred.prob<- predict(svm.cred,log.valid[,-24],decision.values = T, probability =
TRUE)
confusionMatrix(svm.pred,log.valid$default.payment.next.month)



cred1<- read.csv("C:/Users/Gaurav/Downloads/default of credit card clients
(1).csv",header = T,skip=1)
knn.cred<- cred1
knn.cred <- knn.cred[,-1]
mynorm <- function(x){
  return ((x-min(x))/(max(x)-min(x)))
}
knn.cred<-as.data.frame(lapply(knn.cred[,1:23], mynorm))
knn.cred$default<-cred1$default.payment.next.month
ktrain.index <- sample(c(1:dim(knn.cred)[1]), 0.7*dim(knn.cred)[1])
k.train<- knn.cred[ktrain.index,]
k.valid<- knn.cred[-ktrain.index,]

knn.cred.reg <- knnreg(default~., k.train, k=20)
knn.cred.pred2 <- predict(knn.cred.reg, k.valid)
knn.class<- ifelse(knn.cred.pred2<0.5,0,1)
kgain <- gains(as.integer(k.valid$default),knn.cred.pred2, groups=10)
kgain
```

```
plot(c(0,kgain$cume.pct.of.total*sum(as.integer(k.valid$default)))~c(0,kgain$cume.obs),
     xlab="# cases", ylab="Cumulative", main="KNN Lift Chart", type="l")
lines(c(0,sum(as.integer(k.valid$default)))~c(0, dim(k.valid)[1]), lty=2)
confusionMatrix(as.factor(knn.class),as.factor(k.valid$default))
View(k.valid)

library(neuralnet)
n= names(k.train)

f <- as.formula(paste("default~", paste(n[!n %in% "default"], collapse = "+")))
n
f


cred.nn <- neuralnet(f, data = k.train, hidden = 5)
View(cred.nn)
cred.nn$data
cred.nn.pred <- compute(cred.nn,k.valid[,1:23])
cred.nn.pred.class <- ifelse(cred.nn.pred$net.result >=0.5, 1, 0)
confusionMatrix(as.factor(cred.nn.pred.class),as.factor(k.valid$default))
nngain <- gains(as.integer(k.valid$default),cred.nn.pred$net.result, groups=10)

plot(c(0,nngain$cume.pct.of.total*sum(as.integer(k.valid$default)))~c(0,nngain$cume.ob
s),
     xlab="# cases", ylab="Cumulative", main="Neural Net Lift Chart", type="l")
lines(c(0,sum(as.integer(k.valid$default)))~c(0, dim(k.valid)[1]), lty=2)
```