Credit Risk Evaluation—Shallow Models

Liang Hu

This work builds 3 classification models (Naïve Bayes, Decision tree, and kNN) to predict credit risk.

Load libraries and read the credit data. Split the data into 2/3 training data set and 1/3 test data set.

```
#load libraries
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(RWeka)
library(e1071)

#read data
setwd(getwd())
credit <- read.csv("credit-g.csv")

#create training set and test set
set.seed(1991)
trainIndex <- createDataPartition(credit$class, p=0.67, list=F, times=1)
creditTrain <- credit[trainIndex, ]
creditTest <- credit[-trainIndex, ]
```

1. Decision trees

Build decision tree model and test the model with independent test set. The estimated true error is 0.291. From the confusion matrix, 53.5% of "bad" creditors will be predicted as "good", and 18.6% of "good" creditors predicted as "bad". The bank is more sensitive to the first type of error.

```
#decision trees, independent test set
model <- J48(class~., data=creditTrain)
prediction <- predict(model, creditTest)
CM <- confusionMatrix(prediction, creditTest$class)
CM

## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   46   43
##       good  53  188
##
##                Accuracy : 0.7091
##                  95% CI : (0.6568, 0.7575)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.3848
##
##                   Kappa : 0.2868
##  Mcnemar's Test P-Value : 0.3583
##
##             Sensitivity : 0.4646
##             Specificity : 0.8139
##          Pos Pred Value : 0.5169
##          Neg Pred Value : 0.7801
##              Prevalence : 0.3000
##          Detection Rate : 0.1394
##    Detection Prevalence : 0.2697
##       Balanced Accuracy : 0.6392
##
##        'Positive' Class : bad
##
```

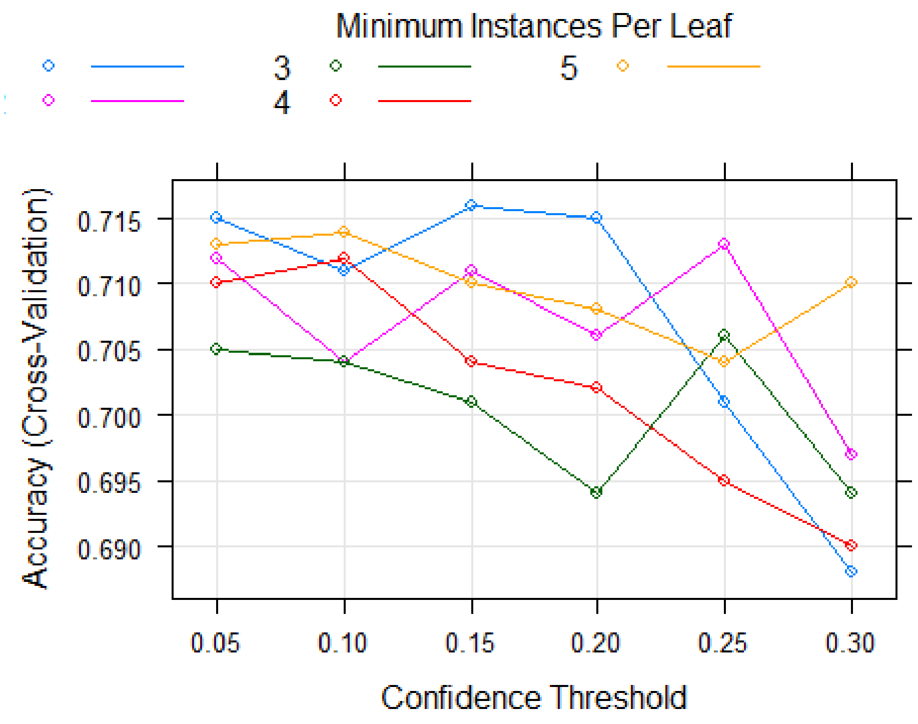
```
print(paste0("The estimated true error is ", 1-round(CM$overall[1], 3)))
```

```
## [1] "The estimated true error is 0.291"
```

Then use cross-validation to estimate the true error. Try different number of folds: 10, 7, and 5. Also, tune parameters C (0.05, 0.10, … , 0.30) and M (1, 2, … , 5). The below table lists the results with the optimal parameter setting. The number of folds has impact on the true error and the parameters. The true error is the smallest when folds equal 7. Note that the estimated true errors using cross-validation are all smaller than using independent test set. Cross-validation tends to over-estimate true error, so for the decision tree model, CV performs better in testing.

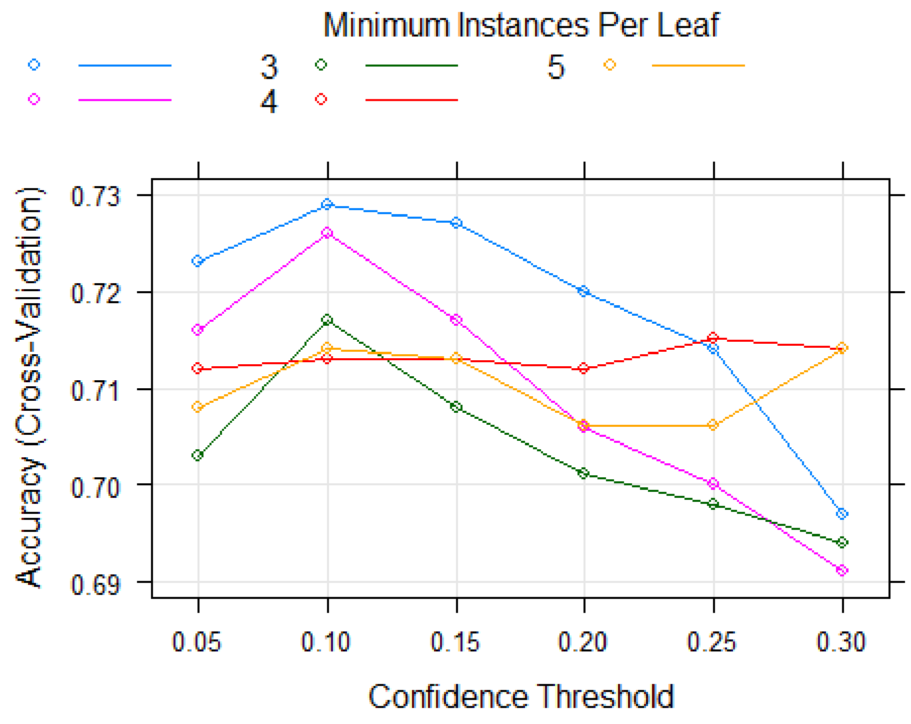
Folds	C	M	True error
10	0.15	1	0.284
7	0.10	1	0.271
5	0.10	4	0.283

```
#decision trees, cross validation (10 folds)
ctrl1 <- trainControl(method="cv", savePred=T, classProb=T, number=10)
#tune parameters
set.seed(1991)
treeGrid <- expand.grid(C=(1:6)*0.05, M=(1:5))
model1 <- train(class~., data=credit, method="J48", trControl=ctrl1,
               tuneGrid=treeGrid)
model1

## C4.5-like Trees
##
## 1000 samples
##   20 predictor
##    2 classes: 'bad', 'good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...
## Resampling results across tuning parameters:
##
##   C     M  Accuracy  Kappa
##   0.05  1  0.715     0.2390455
##   0.05  2  0.712     0.2309407
##   0.05  3  0.705     0.2106388
##   0.05  4  0.710     0.2349606
##   0.05  5  0.713     0.2472694
##   0.10  1  0.711     0.2602618
##   0.10  2  0.704     0.2461223
##   0.10  3  0.704     0.2301517
##   0.10  4  0.712     0.2452380
##   0.10  5  0.714     0.2506770
##   0.15  1  0.716     0.2754365
##   0.15  2  0.711     0.2690253
##   0.15  3  0.701     0.2374045
##   0.15  4  0.704     0.2376227
##   0.15  5  0.710     0.2507128
##   0.20  1  0.715     0.2758136
##   0.20  2  0.706     0.2500395
##   0.20  3  0.694     0.2275633
##   0.20  4  0.702     0.2304679
##   0.20  5  0.708     0.2530078
##   0.25  1  0.701     0.2474762
##   0.25  2  0.713     0.2720056
##   0.25  3  0.706     0.2584877
##   0.25  4  0.695     0.2192740
##   0.25  5  0.704     0.2431222
##   0.30  1  0.688     0.2258259
##   0.30  2  0.697     0.2467477
##   0.30  3  0.694     0.2473556
##   0.30  4  0.690     0.2251806
##   0.30  5  0.710     0.2622210
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were C = 0.15 and M = 1.

plot(model1)
```



```
print("The estimated true error is 0.284.")

## [1] "The estimated true error is 0.284."

print("The selected parameters are C=0.15 and M=1.")

## [1] "The selected parameters are C=0.15 and M=1."

#decision trees, cross validation (7 folds)
ctrl2 <- trainControl(method="cv", savePred=T, classProb=T, number=7)
#tune parameters
set.seed(1991)
treeGrid <- expand.grid(C=(1:6)*0.05, M=(1:5))
model2 <- train(class~., data=credit, method="J48", trControl=ctrl2,
               tuneGrid=treeGrid)
model2

## C4.5-like Trees
##
## 1000 samples
##   20 predictor
##    2 classes: 'bad', 'good'
##
## No pre-processing
## Resampling: Cross-Validated (7 fold)
## Summary of sample sizes: 857, 858, 857, 857, 857, 857, ...
## Resampling results across tuning parameters:
##
##   C     M  Accuracy   Kappa
##   0.05  1  0.7230235  0.2766773
##   0.05  2  0.7160304  0.2187602
##   0.05  3  0.7030223  0.1987023
##   0.05  4  0.7120133  0.2209640
##   0.05  5  0.7080173  0.2131630
##   0.10  1  0.7290245  0.2978795
##   0.10  2  0.7260205  0.2921374
##   0.10  3  0.7170576  0.2692690
##   0.10  4  0.7130546  0.2411762
##   0.10  5  0.7140536  0.2406609
##   0.15  1  0.7270124  0.3009694
##   0.15  2  0.7170083  0.2715789
##   0.15  3  0.7080384  0.2584704
##   0.15  4  0.7130334  0.2499132
##   0.15  5  0.7130264  0.2486213
##   0.20  1  0.7200194  0.2884169
##   0.20  2  0.7059982  0.2525474
##   0.20  3  0.7010384  0.2444040
##   0.20  4  0.7120344  0.2543303
##   0.20  5  0.7060334  0.2386080
##   0.25  1  0.7140113  0.2772686
##   0.25  2  0.7000183  0.2425235
##   0.25  3  0.6980484  0.2470066
##   0.25  4  0.7150314  0.2620255
##   0.25  5  0.7060334  0.2414448
##   0.30  1  0.6970143  0.2447743
##   0.30  2  0.6910484  0.2243059
##   0.30  3  0.6940454  0.2424228
##   0.30  4  0.7140465  0.2769841
##   0.30  5  0.7140324  0.2632994
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were C = 0.1 and M = 1.
```

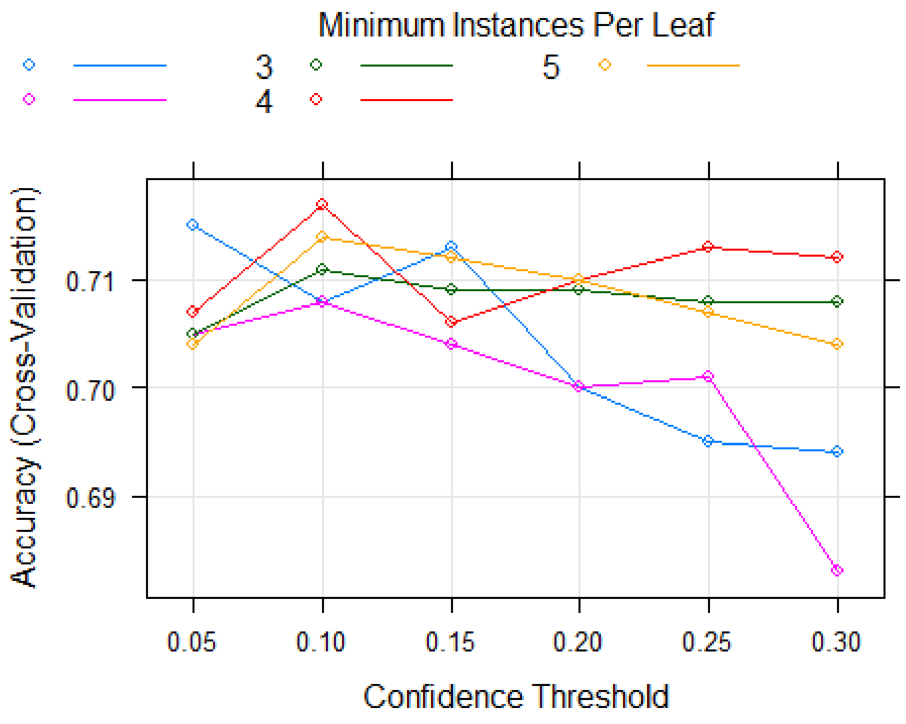
```
plot(model2)
```



```
print("The estimated true error is 0.271.")

## [1] "The estimated true error is 0.271."

print("The selected parameters are C=0.10 and M=1.")

## [1] "The selected parameters are C=0.10 and M=1."

#decision trees, cross validation (5 folds)
ctrl3 <- trainControl(method="cv", savePred=T, classProb=T, number=5)
#tune parameters
set.seed(1991)
treeGrid <- expand.grid(C=(1:6)*0.05, M=(1:5))
model3 <- train(class~., data=credit, method="J48", trControl=ctrl3,
               tuneGrid=treeGrid)
model3

## C4.5-like Trees
##
## 1000 samples
##   20 predictor
##    2 classes: 'bad', 'good'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 800, 800, 800, 800, 800
## Resampling results across tuning parameters:
##
```

```
##   C     M  Accuracy  Kappa
##   0.05  1  0.715     0.2351280
##   0.05  2  0.705     0.2207741
##   0.05  3  0.705     0.2213832
##   0.05  4  0.707     0.2221854
##   0.05  5  0.704     0.2062329
##   0.10  1  0.708     0.2532516
##   0.10  2  0.708     0.2571423
##   0.10  3  0.711     0.2411650
##   0.10  4  0.717     0.2749567
##   0.10  5  0.714     0.2613800
##   0.15  1  0.713     0.2584270
##   0.15  2  0.704     0.2402197
##   0.15  3  0.709     0.2646027
##   0.15  4  0.706     0.2482722
##   0.15  5  0.712     0.2454648
##   0.20  1  0.700     0.2316797
##   0.20  2  0.700     0.2268438
##   0.20  3  0.709     0.2707956
##   0.20  4  0.710     0.2652175
##   0.20  5  0.710     0.2461004
##   0.25  1  0.695     0.2362536
##   0.25  2  0.701     0.2328044
##   0.25  3  0.708     0.2633890
##   0.25  4  0.713     0.2727150
##   0.25  5  0.707     0.2540249
##   0.30  1  0.694     0.2476112
##   0.30  2  0.683     0.2198861
##   0.30  3  0.708     0.2624234
##   0.30  4  0.712     0.2705993
##   0.30  5  0.704     0.2481933
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were C = 0.1 and M = 4.
```

```
plot(model3)
```



```
print("The estimated true error is 0.283.")

## [1] "The estimated true error is 0.283."

print("The selected parameters are C=0.10 and M=4.")

## [1] "The selected parameters are C=0.10 and M=4."
```

2. Naïve Bayes

Build a Naïve Bayes model and test the model with the independent test set. The estimated true error is 0.258. From the confusion matrix, about 50.5% of “bad” creditors will be predicted as “good” by the NB model, and 15% of “good” creditors predicted as “bad”.

```
#naive Bayes, independent test set
model <- naiveBayes(class~., data=creditTrain)
prediction <- predict(model, creditTest)
CM <- confusionMatrix(prediction, creditTest$class)
CM

## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   49   35
##       good  50  196
```

```
##
##                Accuracy : 0.7424
##                  95% CI : (0.6917, 0.7888)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.05091
##
##                   Kappa : 0.359
##  Mcnemar's Test P-Value : 0.12889
##
##             Sensitivity : 0.4949
##             Specificity : 0.8485
##          Pos Pred Value : 0.5833
##          Neg Pred Value : 0.7967
##              Prevalence : 0.3000
##          Detection Rate : 0.1485
##    Detection Prevalence : 0.2545
##       Balanced Accuracy : 0.6717
##
##        'Positive' Class : bad
##

print(paste0("The estimated true error is ", 1-round(CM$overall[1], 3)))

## [1] "The estimated true error is 0.258"
```

Then use cross-validation to estimate the true error. Try different number of folds: 10, 7, and 5. The estimate true error at different folds is all 0.3, which is large than 0.258. Therefore, for the Naïve Bayes model, independent test set performs better in testing the model.

```
#naive Bayes, cross validation (10 folds)
ctrl1 <- trainControl(method="cv", savePred=T, classProb=T, number=10)
set.seed(1991)
model1 <- train(class~., data=credit, method="nb", trControl=ctrl1)

model1

## Naive Bayes
##
## 1000 samples
##   20 predictor
##    2 classes: 'bad', 'good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.6922222  0.3230312
##    TRUE      0.7000000  0.0000000
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were fL = 0, usekernel = TRUE
##  and adjust = 1.
```

```
print("The estimated true error is 0.3.")
```

```
## [1] "The estimated true error is 0.3."
```

```
#naive Bayes, cross validation (7 folds)
ctrl2 <- trainControl(method="cv", savePred=T, classProb=T, number=7)
set.seed(1991)
model2 <- train(class~., data=credit, method="nb", trControl=ctrl2)

model2
```

```
## Naive Bayes
##
## 1000 samples
##   20 predictor
##    2 classes: 'bad', 'good'
##
## No pre-processing
## Resampling: Cross-Validated (7 fold)
## Summary of sample sizes: 857, 858, 857, 857, 857, 857, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.6989560  0.349867905
##    TRUE      0.7000042  0.002600306
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were fL = 0, usekernel = TRUE
##  and adjust = 1.
```

```
print("The estimated true error is 0.3.")
```

```
## [1] "The estimated true error is 0.3."
```

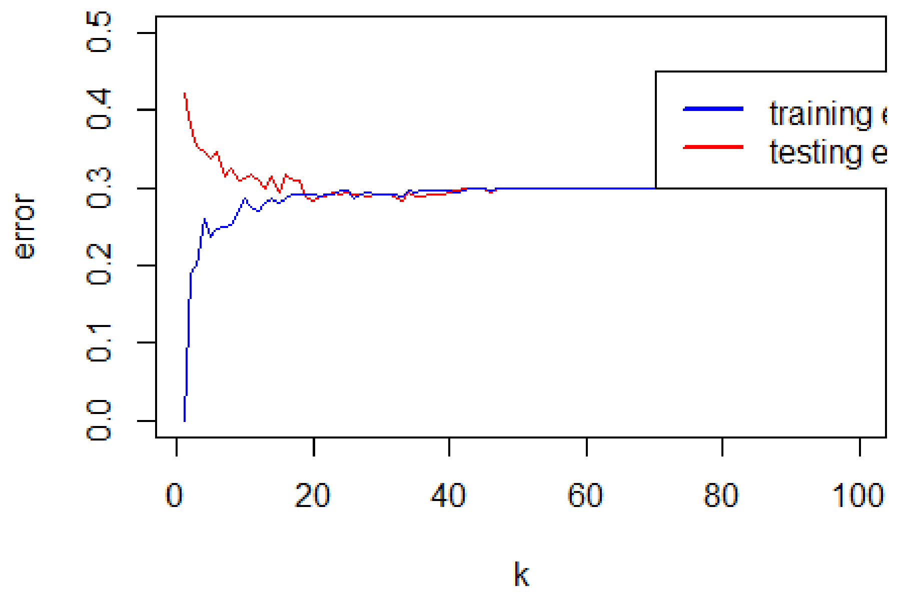
```
#naive Bayes, cross validation (5 folds)
ctrl3 <- trainControl(method="cv", savePred=T, classProb=T, number=5)
set.seed(1991)
model3 <- train(class~., data=credit, method="nb", trControl=ctrl3)
```

```
model3

## Naive Bayes
##
## 1000 samples
##   20 predictor
##    2 classes: 'bad', 'good'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 800, 800, 800, 800, 800
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy  Kappa
##   FALSE      0.69375   0.3348667
##    TRUE      0.70000   0.0000000
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were fL = 0, usekernel = TRUE
##  and adjust = 1.

print("The estimated true error is 0.3.")

## [1] "The estimated true error is 0.3."
```

3. kNN

Build a kNN model and test the model with the independent test set. Try different number of neighbors from 1 to 100. Plot the training errors and test errors. From the error-k plot, it can be seen that when k<10, the model fits the training data well but poorly on the test data. When k>=20, the training error and test error are almost the same. We select k=20 and see the confusion matrix. The prediction accuracy is 0.7091. However, 94 "bad" in 99 "bad" are predicted as "good". This obviously increases the bank's credit risk.

```
#kNN, independent test set
#tune the parameter k
train_error <- c()
test_error <- c()
for(i in 1:100) {
  model <- knn3(class~., data=creditTrain, k=i)
  prediction_test <- predict(model, creditTest, type="class")
  test_error[i] <- 1-confusionMatrix(prediction_test, creditTest$class)$overa
ll[1]
  prediction_train <- predict(model, creditTrain, type="class")
  train_error[i] <- 1-confusionMatrix(prediction_train, creditTrain$class)$ov
erall[1]
}
plot(c(1, 100), c(0, 0.5), type='n', xlab='k', ylab='error')
lines(test_error, col='red')
lines(train_error, col='blue')
legend(70, 0.45, c('training error', 'testing error'),
       lty=c(1,1), lwd=c(2.5,2.5), col=c('blue', 'red'))
```



```
#try k=20
model <- knn3(class~., data=creditTrain, k=20)
prediction <- predict(model, creditTest, type="class")
CM <- confusionMatrix(prediction, creditTest$class)
CM

## Confusion Matrix and Statistics
##
##             Reference
## Prediction bad good
##       bad    5    2
##       good  94  229
##
##                Accuracy : 0.7091
##                  95% CI : (0.6568, 0.7575)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.3848
##
##                   Kappa : 0.057
##  Mcnemar's Test P-Value : <2e-16
```

```
##
##             Sensitivity : 0.05051
##             Specificity : 0.99134
##          Pos Pred Value : 0.71429
##          Neg Pred Value : 0.70898
##              Prevalence : 0.30000
##          Detection Rate : 0.01515
##    Detection Prevalence : 0.02121
##       Balanced Accuracy : 0.52092
##
##        'Positive' Class : bad
##
```

```
print(paste0("The estimated true error is ", 1-round(CM$overall[1], 3)))
```

```
## [1] "The estimated true error is 0.291"
```

Then use cross-validation to estimate the true error. Try different number of folds: 10, 7, and 5. Also, tune parameter k from 1 to 100. The below table lists the results with the optimal parameter setting. The number of folds does not have significant impact on the true error, which stands at about 0.288. The optimal k selected by the algorithm is 69 or 64, but from the table the accuracy is keep almost constant when k>=22. Also, the same problem exists—the majority of "bad" are predicted as "good". This type of error is more serious and increases the bank's credit risk, even though the true error is low. When k=20, the true error estimated by CV is 0.31, which is larger than using independent test data.

Folds	k	True error
10	69	0.288
7	64	0.288
5	64	0.287

```
#kNN, cross validation (10 folds)
set.seed(1991)
model1 <- train(class~., data=credit, method="knn", trControl=ctrl1,
               tuneGrid=expand.grid(k=c(1:100)))
model1
```

```
## k-Nearest Neighbors
##
## 1000 samples
##   20 predictor
##    2 classes: 'bad', 'good'
```

```
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...
## Resampling results across tuning parameters:
##
##    k    Accuracy  Kappa
##     1  0.622     0.092321926
##     2  0.593     0.017851333
##     3  0.622     0.037655319
##     4  0.643     0.093924273
##     5  0.663     0.096428435
##     6  0.653     0.079092322
##     7  0.674     0.098641581
##     8  0.668     0.077927924
##     9  0.678     0.080648313
##    10  0.673     0.071967590
##    11  0.680     0.075247386
##    12  0.678     0.054359960
##    13  0.691     0.087409010
##    14  0.696     0.094995995
##    15  0.698     0.086963477
##    16  0.694     0.084239287
##    17  0.700     0.085936137
##    18  0.703     0.096682992
##    19  0.697     0.075049118
##    20  0.690     0.057607391
##    21  0.696     0.064074660
##    22  0.701     0.075813632
##    23  0.705     0.083172732
##    24  0.704     0.080233615
##    25  0.708     0.081588997
##    26  0.706     0.073140294
##    27  0.708     0.074058028
##    28  0.708     0.080869012
##    29  0.705     0.070365091
##    30  0.707     0.069261249
##    31  0.708     0.073769595
##    32  0.710     0.075624571
##    33  0.709     0.071391976
##    34  0.710     0.073399836
##    35  0.710     0.070863707
##    36  0.709     0.067136168
##    37  0.709     0.067021151
##    38  0.709     0.069150517
##    39  0.712     0.075198929
##    40  0.711     0.070695521
##    41  0.709     0.066963574
##    42  0.707     0.063123442
##    43  0.708     0.062546317
##    44  0.709     0.069534649
##    45  0.711     0.075805737
##    46  0.707     0.065659572
##    47  0.710     0.073745124
##    48  0.706     0.059276534
##    49  0.708     0.067776927
##    50  0.705     0.055239174
##    51  0.704     0.048267005
##    52  0.708     0.063047744
##    53  0.709     0.060465619
##    54  0.708     0.065229495
##    55  0.710     0.071055628
##    56  0.710     0.069120175
##    57  0.711     0.073100381
##    58  0.709     0.066552220
##    59  0.709     0.066552220
##    60  0.710     0.071085042
##    61  0.710     0.068596973
##    62  0.711     0.070695521
##    63  0.710     0.068596973
##    64  0.712     0.072683512
##    65  0.712     0.072683512
##    66  0.711     0.070638759
##    67  0.711     0.068450917
##    68  0.710     0.066381783
##    69  0.712     0.072683512
##    70  0.711     0.062188456
##    71  0.709     0.053040655
##    72  0.706     0.040066801
##    73  0.707     0.044437430
##    74  0.709     0.058017677
##    75  0.706     0.037650288
##    76  0.705     0.033164642
##    77  0.702     0.019583487
##    78  0.703     0.024188750
##    79  0.702     0.019583487
##    80  0.703     0.024188750
##    81  0.702     0.014354067
##    82  0.700     0.005263158
##    83  0.702     0.009210526
##    84  0.701     0.004605263
##    85  0.700     0.000000000
##    86  0.700     0.000000000
##    87  0.700     0.000000000
##    88  0.700     0.000000000
##    89  0.700     0.000000000
##    90  0.700     0.000000000
##    91  0.700     0.000000000
##    92  0.700     0.000000000
##    93  0.700     0.000000000
##    94  0.700     0.000000000
##    95  0.700     0.000000000
##    96  0.700     0.000000000
##    97  0.700     0.000000000
##    98  0.700     0.000000000
##    99  0.700     0.000000000
##   100  0.700     0.000000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 69.
```

```
print("The estimated true error is 0.288.")
```

```
## [1] "The estimated true error is 0.288."
```

```
print("The selected parameters are k=69.")
```

```
## [1] "The selected parameters are k=69."
```

```
#kNN, cross validation (7 folds)
set.seed(1991)
model2 <- train(class~., data=credit, method="knn", trControl=ctrl2,
               tuneGrid=expand.grid(k=c(1:100)))
model2
```

```
## k-Nearest Neighbors
##
## 1000 samples
##   20 predictor
##    2 classes: 'bad', 'good'
##
## No pre-processing
## Resampling: Cross-Validated (7 fold)
## Summary of sample sizes: 857, 858, 857, 857, 857, 857, ...
## Resampling results across tuning parameters:
##
##   k    Accuracy   Kappa
##     1  0.6139916   0.0690621486
##     2  0.5929352   0.0254685855
##     3  0.6229967   0.0425629717
##     4  0.6420481   0.0957846925
##     5  0.6590170   0.0939172098
##     6  0.6520170   0.0628683308
##     7  0.6710051   0.0954493416
##     8  0.6649970   0.0683847251
##     9  0.6819870   0.0957319783
##    10  0.6840061   0.1002106900
##    11  0.6859901   0.0763484641
##    12  0.6879810   0.0711884369
##    13  0.6919841   0.0803656503
##    14  0.6799961   0.0544647819
##     15  0.6899931   0.0665409434
##     16  0.6869961   0.0620443715
##     17  0.6949951   0.0710617808
##     18  0.6949951   0.0752467387
##     19  0.6909921   0.0585707343
##     20  0.6969931   0.0658669450
##     21  0.7000042   0.0688736107
##     22  0.7010032   0.0731567575
##     23  0.7049992   0.0786347802
##     24  0.7039932   0.0769362475
##     25  0.7040073   0.0699082338
##     26  0.7089952   0.0865488097
##     27  0.7059982   0.0712226050
##     28  0.7069972   0.0753561899
##     29  0.7100013   0.0788732861
##     30  0.7110003   0.0858554112
##     31  0.7080033   0.0750764769
##     32  0.7099942   0.0765948145
##     33  0.7100013   0.0722207195
##     34  0.7070043   0.0663338382
##     35  0.7080033   0.0637648423
##     36  0.7080033   0.0637249897
##     37  0.7100013   0.0698613795
##     38  0.7100013   0.0698613795
##     39  0.7100013   0.0698613795
##     40  0.7100013   0.0698477068
##     41  0.7070043   0.0618433666
##     42  0.7059982   0.0598244223
##     43  0.7110003   0.0719823782
##     44  0.7069972   0.0663956275
##     45  0.7079962   0.0659256950
##     46  0.7079962   0.0682119993
##     47  0.7080033   0.0658979732
##     48  0.7090093   0.0705666829
##     49  0.7070043   0.0638334198
##     50  0.7069972   0.0637906564
##     51  0.7080103   0.0638731265
##     52  0.7070043   0.0565938811
##     53  0.7070113   0.0592363606
##     54  0.7100083   0.0703216992
##     55  0.7100083   0.0651861585
##     56  0.7090093   0.0632065074
##     57  0.7100083   0.0651861585
##     58  0.7110073   0.0697899981
##     59  0.7100083   0.0677864641
##     60  0.7120063   0.0767007533
##     61  0.7100083   0.0651861585
##     62  0.7120063   0.0743100379
##     63  0.7110073   0.0697899981
##     64  0.7120063   0.0743100379
##    65  0.7100083   0.0651861585
##    66  0.7110073   0.0697899981
##    67  0.7090023   0.0605321704
##    68  0.7110003   0.0673493097
##    69  0.7110073   0.0672181029
##    70  0.7080033   0.0540438756
##    71  0.7059982   0.0446955278
##    72  0.7069972   0.0466277515
##    73  0.7090023   0.0507393483
##    74  0.7060053   0.0423991345
##    75  0.7080033   0.0440099483
##    76  0.7049992   0.0276013475
##    77  0.7059982   0.0297623538
##    78  0.7049992   0.0226862665
##    79  0.7049992   0.0226862665
##    80  0.7040002   0.0182477589
##    81  0.7049992   0.0226862665
##    82  0.7040002   0.0182477589
##    83  0.7010032   0.0046038396
##    84  0.7010032   0.0046038396
##    85  0.7000042   0.0000000000
##    86  0.7000042   0.0000000000
##    87  0.7000042   0.0000000000
##    88  0.7000042   0.0000000000
##    89  0.7000042   0.0000000000
##    90  0.6990052   0.0006329114
##    91  0.6980062  -0.0012993123
##    92  0.7000042   0.0026003056
##    93  0.7000042   0.0000000000
##    94  0.7010032   0.0046038396
##    95  0.7020022   0.0091238795
##    96  0.7000042   0.0000000000
##    97  0.7000042   0.0000000000
##    98  0.7000042   0.0000000000
##    99  0.7000042   0.0000000000
##   100  0.7000042   0.0000000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 64.
```

```
print("The estimated true error is 0.288.")
```

```
## [1] "The estimated true error is 0.288."
```

```
print("The selected parameters are k=64.")
```

```
## [1] "The selected parameters are k=64."
```

```
#kNN, cross validation (5 folds)
set.seed(1991)
model3 <- train(class~., data=credit, method="knn", trControl=ctrl3,
               tuneGrid=expand.grid(k=c(1:100)))
model3

## k-Nearest Neighbors
##
## 1000 samples
##   20 predictor
##    2 classes: 'bad', 'good'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 800, 800, 800, 800, 800
## Resampling results across tuning parameters:
##
##   k    Accuracy  Kappa
##     1  0.606     0.0683001574
##     2  0.607     0.0463372352
##     3  0.634     0.0731574020
##     4  0.638     0.0891727302
##     5  0.655     0.0797401366
##     6  0.653     0.0594160371
##     7  0.665     0.0714954253
##     8  0.658     0.0482731649
##     9  0.677     0.0749247525
##    10  0.679     0.0979726860
##    11  0.686     0.0897341656
##    12  0.684     0.0850352426
##    13  0.697     0.1016174188
##    14  0.699     0.1119097062
##    15  0.703     0.1088979459
##    16  0.699     0.0917963015
##    17  0.703     0.0954125175
##    18  0.699     0.0877562614
##    19  0.703     0.0881149640
##    20  0.698     0.0697282137
##    21  0.708     0.0978090740
##    22  0.708     0.0998010232
##    23  0.706     0.0847538511
##    24  0.712     0.1010841288
##    25  0.706     0.0692351058
##    26  0.708     0.0780081251
##    27  0.710     0.0794769939
##    28  0.707     0.0712298714
##    29  0.710     0.0773199520
##    30  0.708     0.0710898938
##    31  0.709     0.0753351768
##    32  0.711     0.0746053703
##    33  0.710     0.0703532406
##    34  0.705     0.0579669953
##    35  0.707     0.0644133384
##     36  0.710      0.0703388110
##     37  0.708      0.0664036258
##     38  0.707      0.0596924862
##     39  0.708      0.0687712674
##     40  0.705      0.0582106354
##     41  0.707      0.0644722466
##     42  0.706      0.0646954436
##     43  0.709      0.0707727127
##     44  0.705      0.0577645734
##     45  0.704      0.0484349808
##     46  0.704      0.0484065364
##     47  0.703      0.0439786528
##     48  0.704      0.0485859700
##     49  0.706      0.0549709384
##     50  0.706      0.0574961684
##     51  0.709      0.0707260582
##     52  0.708      0.0639477813
##     53  0.711      0.0770297960
##     54  0.711      0.0770297960
##     55  0.711      0.0745933879
##     56  0.707      0.0519457421
##     57  0.709      0.0680387870
##     58  0.709      0.0680387870
##     59  0.710      0.0700688801
##     60  0.711      0.0723063835
##     61  0.712      0.0741481342
##     62  0.712      0.0741481342
##     63  0.710      0.0700688801
##     64  0.713      0.0740469572
##     65  0.708      0.0490799447
##     66  0.705      0.0385366268
##     67  0.708      0.0539949030
##     68  0.704      0.0339200273
##     69  0.704      0.0287252221
##     70  0.705      0.0307294048
##     71  0.703      0.0216373152
##     72  0.701      0.0124267889
##     73  0.701      0.0124267889
##     74  0.702      0.0143790850
##     75  0.703      0.0163570691
##     76  0.701      0.0072237713
##     77  0.700      0.0026315789
##     78  0.700      0.0052631579
##     79  0.700      0.0000000000
##     80  0.700      0.0052631579
##     81  0.700      0.0000000000
##     82  0.700      0.0000000000
##     83  0.700      0.0000000000
##     84  0.700      0.0000000000
##     85  0.700      0.0000000000
##    86  0.700     0.0000000000
##    87  0.700     0.0000000000
##    88  0.702     0.0092105263
##    89  0.700     0.0000000000
##    90  0.704     0.0183612518
##    91  0.703     0.0137254902
##    92  0.705     0.0228175798
##    93  0.702     0.0117387352
##    94  0.699     0.0006883932
##    95  0.700     0.0026315789
##    96  0.700     0.0000000000
##    97  0.700     0.0000000000
##    98  0.700     0.0000000000
##    99  0.700     0.0000000000
##   100  0.700     0.0000000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 64.
```

```
print("The estimated true error is 0.287.")
```

```
## [1] "The estimated true error is 0.287."
```

```
print("The selected parameters are k=64.")
```

```
## [1] "The selected parameters are k=64."
```

4. Summary

Cross-validation generally results in smaller true error estimation than independent test set. The Naïve Bayes model trained with 2/3 dataset has the highest accuracy, but about half of bad creditors are predicted as good, which increases the bank's credit risk. Also, the assumption that the creditors' attributes are independent is hardly satisfied. The decision tree model also has high accuracy, but it suffers from the similar problem—the error of bad creditors being predicted as good is about 50%. The kNN model performs the worst; the accuracy is lower than the other 2 models, and majority of bad creditors are predicted as good.