Credit Risk Evaluation—Advanced Classification Models

Liang Hu

Load libraries.

```
library(caret)

library(RWeka)
library(e1071)
library(DMwR)

library(ggplot2)
```

Read data.

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

Use SMOTE to improve minority class prediction.

```
set.seed(1991)
credit_smote <- SMOTE(class~., data=credit)
table(credit_smote$class)

##
##  bad good
##  900 1200
```

Use 10-fold cross-validation to evaluate error.

```
ctrl_cv10 <- trainControl(method='cv', number=10, savePredictions=T,
                          classProbs=T)
```

1. Random Forest Model

Build the random forest model. The model accuracy estimated by 10-fold cross-validation is 0.9033 with 25 mtry. The confusion matrix shows that the classes can be very well predicted.

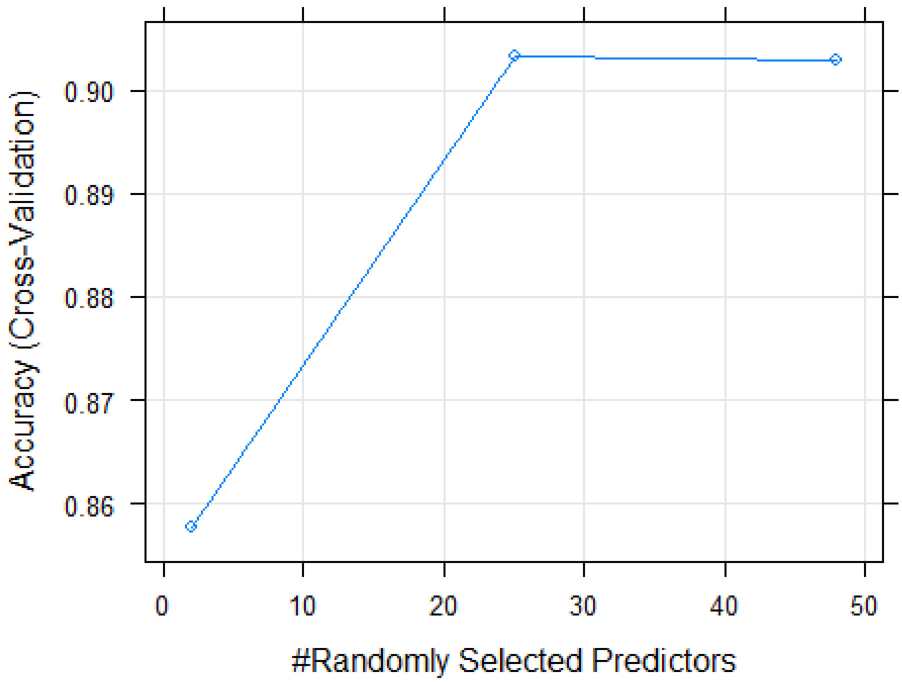
```
model_RF <- train(class~., data=credit_smote, method='rf',
trControl=ctrl_cv10)
model_RF
```

```
## Random Forest
##
## 2100 samples
##   20 predictor
##    2 classes: 'bad', 'good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1890, 1890, 1890, 1890, 1890, 1890, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8576190  0.6992317
##   25    0.9033333  0.8006711
##   48    0.9028571  0.7988768
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 25.
```

```
prediction_RF <- predict(model_RF, credit_smote)
confusionMatrix(prediction_RF, credit_smote$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  bad good
##       bad   900    0
##       good    0 1200
##
##                Accuracy : 1
##                  95% CI : (0.9982, 1)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4286
##          Detection Rate : 0.4286
##    Detection Prevalence : 0.4286
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : bad
##
```

```
plot(model_RF)
```



2. SVM model with different kernels

2.1. SVM with linear kernel

Build the SVM model using linear kernel without specifying parameters. When the cost equals 1, the model has the highest accuracy of 0.7657. The confusion matrix shows that 28.4% bad creditors are predicted as good.

```
model_SVM1 <- train(class~., data=credit_smote, method='svmLinear2',
trControl=ctrl_cv10)
model_SVM1

## Support Vector Machines with Linear Kernel
##
## 2100 samples
##   20 predictor
##    2 classes: 'bad', 'good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1890, 1890, 1890, 1890, 1890, 1890, ...
## Resampling results across tuning parameters:
##
##   cost  Accuracy   Kappa
##   0.25  0.7628571  0.5121175
```

```
##   0.50  0.7652381  0.5172171
##   1.00  0.7657143  0.5185264
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cost = 1.
```

```
prediction_SVM1 <- predict(model_SVM1, credit_smote)
confusionMatrix(prediction_SVM1, credit_smote$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad  644  202
##       good 256  998
##
##                Accuracy : 0.7819
##                  95% CI : (0.7636, 0.7994)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.5514
##  Mcnemar's Test P-Value : 0.01327
##
##             Sensitivity : 0.7156
##             Specificity : 0.8317
##          Pos Pred Value : 0.7612
##          Neg Pred Value : 0.7959
##              Prevalence : 0.4286
##          Detection Rate : 0.3067
##    Detection Prevalence : 0.4029
##       Balanced Accuracy : 0.7736
##
##        'Positive' Class : bad
##
```

Then, we tune the parameters *cost* of the SVM model using linear kernel from 0.1 to 1. When *cost* equals 0.2, the model has the highest accuracy of 0.771. Moreover, the confusion matrix shows that 28.4% bad creditors are predicted as good.

```
model_SVM2 <- train(class~., data=credit_smote, method='svmLinear2',
trControl=ctrl_cv10, tuneGrid=expand.grid(cost=0.1*c(1:10)))
model_SVM2
```

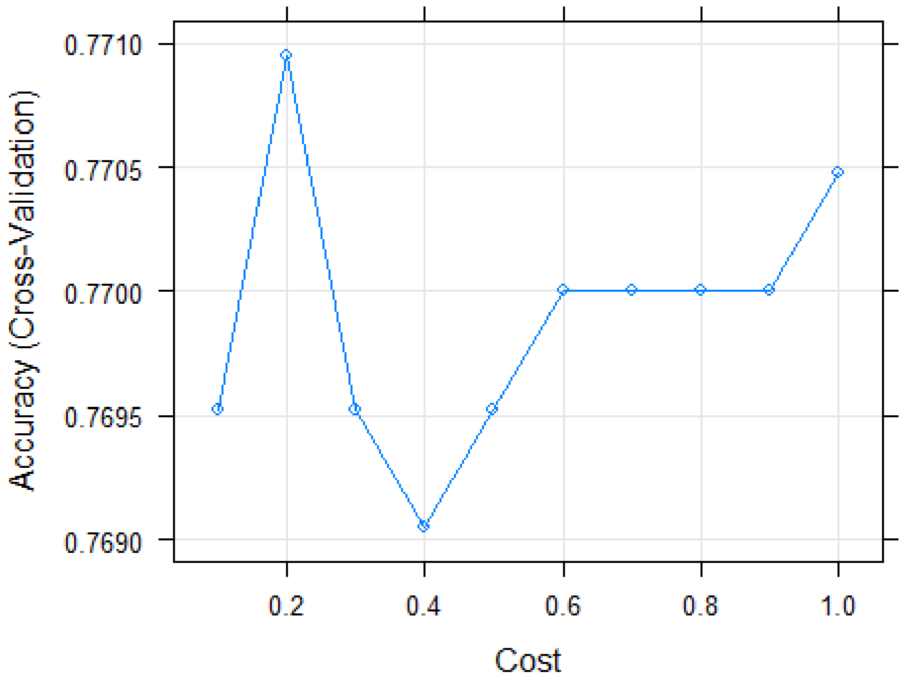
```
## Support Vector Machines with Linear Kernel
##
## 2100 samples
##   20 predictor
##    2 classes: 'bad', 'good'
```

```
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1890, 1890, 1890, 1890, 1890, 1890, ...
## Resampling results across tuning parameters:
##
##   cost  Accuracy   Kappa
##   0.1   0.7695238  0.5254958
##   0.2   0.7709524  0.5287593
##   0.3   0.7695238  0.5257874
##   0.4   0.7690476  0.5249200
##   0.5   0.7695238  0.5258176
##   0.6   0.7700000  0.5269908
##   0.7   0.7700000  0.5269908
##   0.8   0.7700000  0.5269908
##   0.9   0.7700000  0.5269908
##   1.0   0.7704762  0.5280334
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cost = 0.2.
```

```
prediction_SVM2 <- predict(model_SVM2, credit_smote)
confusionMatrix(prediction_SVM2, credit_smote$class)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction bad good
##       bad  644  202
##       good 256  998
##
##                Accuracy : 0.7819
##                  95% CI : (0.7636, 0.7994)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.5514
##  Mcnemar's Test P-Value : 0.01327
##
##             Sensitivity : 0.7156
##             Specificity : 0.8317
##          Pos Pred Value : 0.7612
##          Neg Pred Value : 0.7959
##              Prevalence : 0.4286
##          Detection Rate : 0.3067
##    Detection Prevalence : 0.4029
##       Balanced Accuracy : 0.7736
##
##        'Positive' Class : bad
##
```

```r
plot(model_SVM2)
```



2.2. SVM model with polynomial kernel

Build the SVM model using polynomial kernel without specifying parameters. The results show that when *degree* = 3, *scale* = 0.1 and *C* = 0.25, the model has the highest accuracy of 0.8852. The confusion matrix shows that bad creditors can be well predicted.

```r
model_SVM3 <- train(class~., data=credit_smote, method='svmPoly',
trControl=ctrl_cv10)
model_SVM3

## Support Vector Machines with Polynomial Kernel
##
## 2100 samples
##   20 predictor
##    2 classes: 'bad', 'good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1890, 1890, 1890, 1890, 1890, 1890, ...
## Resampling results across tuning parameters:
##
##   degree  scale  C     Accuracy   Kappa
##   1       0.001  0.25  0.7414286  0.4866490
##   1       0.001  0.50  0.7514286  0.5000923
##   1       0.001  1.00  0.7638095  0.5169157
```

```
##   1       0.010  0.25  0.7695238  0.5264142
##   1       0.010  0.50  0.7761905  0.5398592
##   1       0.010  1.00  0.7700000  0.5267170
##   1       0.100  0.25  0.7714286  0.5291652
##   1       0.100  0.50  0.7709524  0.5270994
##   1       0.100  1.00  0.7714286  0.5284754
##   2       0.001  0.25  0.7533333  0.5040280
##   2       0.001  0.50  0.7638095  0.5169109
##   2       0.001  1.00  0.7680952  0.5240473
##   2       0.010  0.25  0.7933333  0.5748905
##   2       0.010  0.50  0.8038095  0.5966047
##   2       0.010  1.00  0.8171429  0.6242931
##   2       0.100  0.25  0.8695238  0.7304538
##   2       0.100  0.50  0.8633333  0.7164726
##   2       0.100  1.00  0.8552381  0.6998669
##   3       0.001  0.25  0.7638095  0.5196234
##   3       0.001  0.50  0.7619048  0.5118752
##   3       0.001  1.00  0.7761905  0.5399881
##   3       0.010  0.25  0.8166667  0.6234494
##   3       0.010  0.50  0.8309524  0.6536944
##   3       0.010  1.00  0.8419048  0.6753399
##   3       0.100  0.25  0.8852381  0.7633923
##   3       0.100  0.50  0.8852381  0.7633320
##   3       0.100  1.00  0.8838095  0.7602588
## 
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were degree = 3, scale = 0.1 and C
##  = 0.25.
```

```
prediction_SVM3 <- predict(model_SVM3, credit_smote)
confusionMatrix(prediction_SVM3, credit_smote$class)
```

```
## Confusion Matrix and Statistics
## 
##             Reference
## Prediction   bad good
##       bad    900    0
##       good     0 1200
## 
##                Accuracy : 1
##                  95% CI : (0.9982, 1)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
## 
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
```

```
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4286
##          Detection Rate : 0.4286
##    Detection Prevalence : 0.4286
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : bad
##
```

Then, we tune the parameters *degree, scale and C* of the SVM model using polynomial kernel. When *degree* = 8, *scale* = 0.1 and *C* = 0.5, the model has the highest accuracy of 0.9043. In the confusion matrix, bad creditors can be well predicted.

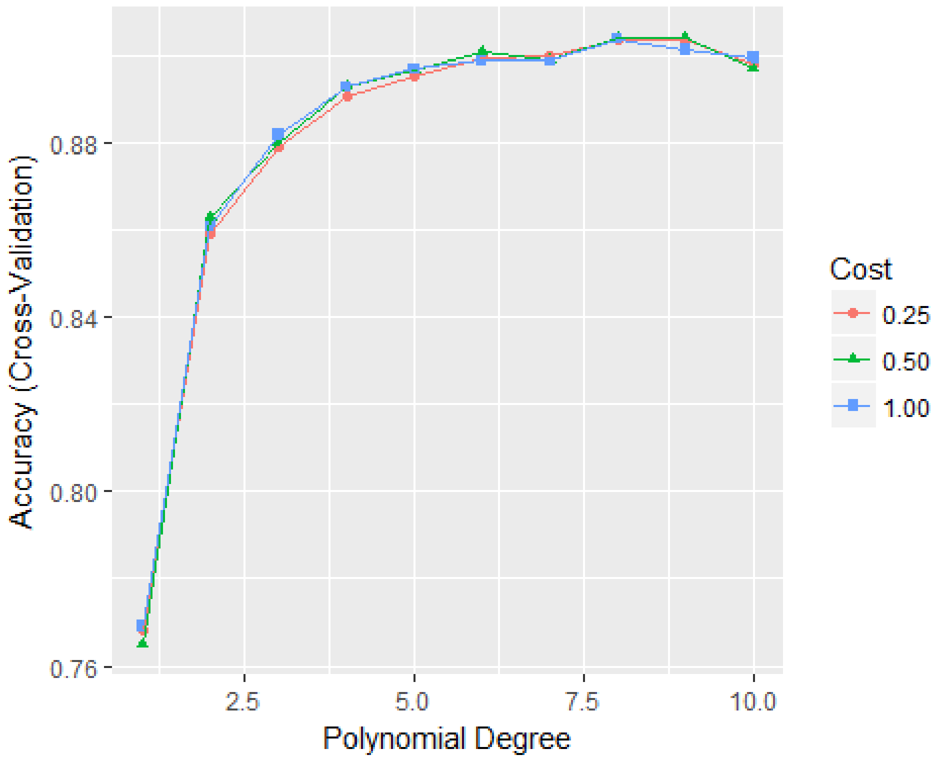
```
svmGrid <- expand.grid(degree=(1:10),
                       scale=0.1,
                       C=c(0.25,0.5,1))
model_SVM4 <- train(class~., data=credit_smote, method='svmPoly',
trControl=ctrl_cv10, tuneGrid=svmGrid)

model_SVM4

## Support Vector Machines with Polynomial Kernel
##
## 2100 samples
##   20 predictor
##    2 classes: 'bad', 'good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1890, 1890, 1890, 1890, 1890, 1890, ...
## Resampling results across tuning parameters:
##
##   degree  C     Accuracy   Kappa
##    1      0.25  0.7680952  0.5217103
##    1      0.50  0.7647619  0.5145067
##    1      1.00  0.7690476  0.5234533
##    2      0.25  0.8590476  0.7079307
##    2      0.50  0.8628571  0.7150561
##    2      1.00  0.8608466  0.7107283
##    3      0.25  0.8790476  0.7502075
##    3      0.50  0.8800000  0.7521438
##    3      1.00  0.8819048  0.7562264
##    4      0.25  0.8909524  0.7761615
##    4      0.50  0.8928571  0.7803041
##    4      1.00  0.8928571  0.7802945
##    5      0.25  0.8952381  0.7853998
##    5      0.50  0.8966667  0.7884795
##    5      1.00  0.8971429  0.7895060
##    6      0.25  0.8995238  0.7950481
```

```
##    6      0.50  0.9009524  0.7980282
##    6      1.00  0.8990476  0.7940008
##    7      0.25  0.9000000  0.7969326
##    7      0.50  0.8990476  0.7946730
##    7      1.00  0.8990476  0.7946252
##    8      0.25  0.9038095  0.8053562
##    8      0.50  0.9042857  0.8060763
##    8      1.00  0.9038095  0.8052260
##    9      0.25  0.9038095  0.8057422
##    9      0.50  0.9042857  0.8067862
##    9      1.00  0.9014286  0.8010898
##   10      0.25  0.8980952  0.7949922
##   10      0.50  0.8971429  0.7934415
##   10      1.00  0.8995238  0.7981320
##
## Tuning parameter 'scale' was held constant at a value of 0.1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were degree = 8, scale = 0.1 and C
##  = 0.5.
```

```
prediction_SVM4 <- predict(model_SVM4, credit_smote)
confusionMatrix(prediction_SVM4, credit_smote$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  bad good
##       bad   900    0
##       good    0 1200
##
##                Accuracy : 1
##                  95% CI : (0.9982, 1)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4286
##          Detection Rate : 0.4286
##    Detection Prevalence : 0.4286
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : bad
##
```

```r
ggplot(model_SVM4)
```



2.3. SVM model with RBF kernel

Build the SVM model using RBF kernel without specifying *sigma*. The model accuracy with *sigma* = 0.0126 is 0.833. The confusion matrix shows that 14.6% of bad creditors are predicted as good.

```r
model_SVM5 <- train(class~., data=credit_smote, method='svmRadial',
trControl=ctrl_cv10)
model_SVM5

## Support Vector Machines with Radial Basis Function Kernel
##
## 2100 samples
##   20 predictor
##    2 classes: 'bad', 'good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1890, 1890, 1890, 1890, 1890, 1890, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.8000000  0.5893515
##   0.50  0.8119048  0.6142252
##   1.00  0.8333333  0.6589182
```

```
##
## Tuning parameter 'sigma' was held constant at a value of 0.01264621
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.01264621 and C = 1.
```

```
prediction_SVM5 <- predict(model_SVM5, credit_smote)
confusionMatrix(prediction_SVM5, credit_smote$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  bad good
##       bad   769   86
##       good  131 1114
##
##                Accuracy : 0.8967
##                  95% CI : (0.8829, 0.9094)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7877
##  Mcnemar's Test P-Value : 0.002818
##
##             Sensitivity : 0.8544
##             Specificity : 0.9283
##          Pos Pred Value : 0.8994
##          Neg Pred Value : 0.8948
##              Prevalence : 0.4286
##          Detection Rate : 0.3662
##    Detection Prevalence : 0.4071
##       Balanced Accuracy : 0.8914
##
##        'Positive' Class : bad
##
```

Tune the parameter *sigma* from 0.01 to 0.02 with an interval of 0.001. The model has the highest accuracy of 0.859 when *sigma* = 0.02. In the confusion matrix, 9.8% of bad creditors are predicted as good.

```
model_SVM6 <- train(class~., data=credit_smote, method='svmRadial',
trControl=ctrl_cv10, tuneGrid=expand.grid(sigma=0.01*seq(1,2,0.1),
                                          C=c(1)))
model_SVM6
```

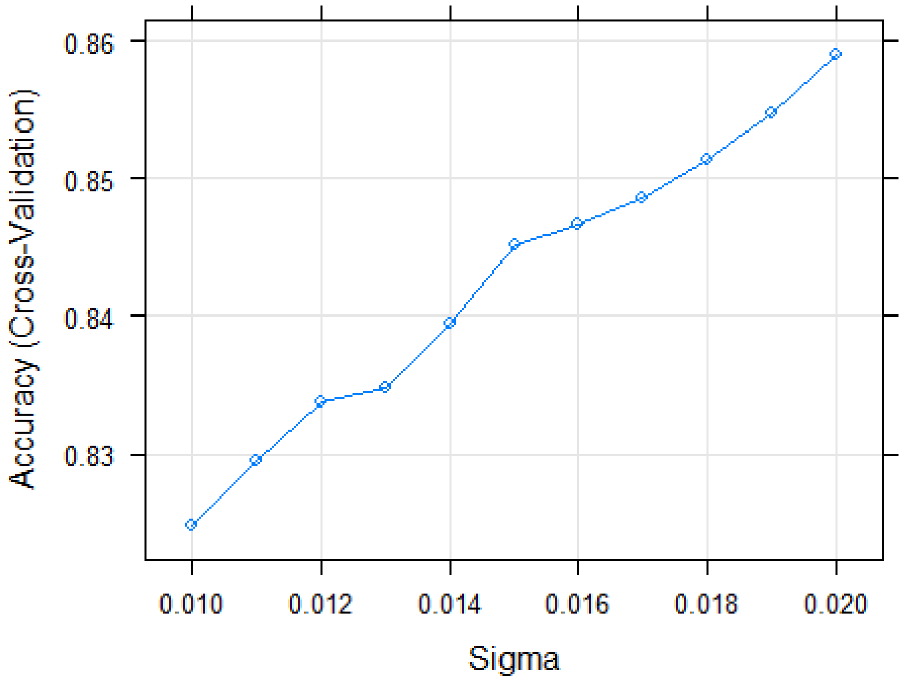
```
## Support Vector Machines with Radial Basis Function Kernel
##
## 2100 samples
##   20 predictor
##    2 classes: 'bad', 'good'
```

```
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1890, 1890, 1890, 1890, 1890, 1890, ...
## Resampling results across tuning parameters:
##
##   sigma  Accuracy   Kappa
##   0.010  0.8247619  0.6410194
##   0.011  0.8295238  0.6508999
##   0.012  0.8338095  0.6597634
##   0.013  0.8347619  0.6619901
##   0.014  0.8395238  0.6713625
##   0.015  0.8452381  0.6828878
##   0.016  0.8466667  0.6859691
##   0.017  0.8485714  0.6899528
##   0.018  0.8514286  0.6958072
##   0.019  0.8547619  0.7024486
##   0.020  0.8590476  0.7113536
##
## Tuning parameter 'C' 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 sigma = 0.02 and C = 1.
```

```
prediction_SVM6 <- predict(model_SVM6, credit_smote)
confusionMatrix(prediction_SVM6, credit_smote$class)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction  bad good
##       bad   812   46
##       good   88 1154
##
##                Accuracy : 0.9362
##                  95% CI : (0.9249, 0.9463)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.869
##  Mcnemar's Test P-Value : 0.0003973
##
##             Sensitivity : 0.9022
##             Specificity : 0.9617
##          Pos Pred Value : 0.9464
##          Neg Pred Value : 0.9291
##              Prevalence : 0.4286
##          Detection Rate : 0.3867
##    Detection Prevalence : 0.4086
##       Balanced Accuracy : 0.9319
##
```

```
##         'Positive' Class : bad
##

plot(model_SVM6)
```



3. Summary and Comparison with Decision Tree, Naïve Bayes, and KNN

By comparing the random forest model, SVM models with different kernels, and the other 3 simple classification models in the table below, it is found that the random forest model and SVM model with polynomial kernel have the highest accuracy and can predict bas creditors very well.

Table 1. Model comparison

Model	Accuracy	Percent of bad creditors predicted as good
Random forest (mtry=25)	0.903	0
SVM (polynomial kernel, degree=8, scale=0.1, C=0.5)	0.904	0
SVM (RBF kernel, C=1, sigma=0.02)	0.859	9.8%

SVM (linear kernel, cost=0.2)	0.771	28.4%
Decision tree (C=0.14, M=5)	0.725	10.8%
Naïve Bayes (fL=1, UseKernal=True, Adjust=4)	0.702	29.9%
kNN (k=43)	0.713	27..8%