## KE5206 | Computational Intelligence I CA2

***Support Vector Machines***

## Data & Package Imports

library(ISLR)  
library(e1071)  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(caTools)  
data(Default)  
summary(Default)

## default student balance income   
## No :9667 No :7056 Min. : 0.0 Min. : 772   
## Yes: 333 Yes:2944 1st Qu.: 481.7 1st Qu.:21340   
## Median : 823.6 Median :34553   
## Mean : 835.4 Mean :33517   
## 3rd Qu.:1166.3 3rd Qu.:43808   
## Max. :2654.3 Max. :73554

We import the necessary packages in order to complete this assignment. The *Default* dataset is taken from the *ISLR* package. We will also deploy SVM from the *e1071* package, to train our model.

*caret* and *caTools* are essential packages used to perform various data processes.

From the summary of *Default*, there are 3 independent variables (binary variable: *student*, numerical variables *balance* and *income*) and one dependent variable (*default*). We will build a model to predict the *default* variable.

## Train-Test Split

We randomly pick 80% of the dataset to be the training set, and the rest as the validation set.

set.seed(111)  
split = sample.split(Default$default, SplitRatio = 0.8)  
  
train = subset(Default, split == T)  
test = subset(Default, split == F)  
  
# train and test sets with balance and income only  
train\_bi = train[, c('default', 'balance', 'income')]  
test\_bi = test[, c('default', 'balance', 'income')]  
  
# train and test sets with student and balance only  
train\_bs = train[, c('default', 'balance', 'student')]  
test\_bs = test[, c('default', 'balance', 'student')]  
  
# train and test sets with student and balance only  
train\_is = train[, c('default', 'income', 'student')]  
test\_is = test[, c('default', 'income', 'student')]

In addition, we have created subsets of the training and test sets by only selecting 2 variables each.

These subsets will be trained on SVM using both radial and sigmoid kernels. These models will also be tuned to determine the best parameters (cost and gamma) to be used, to optimize training.

## Balance & Income Dataset (BI)

### Fitting Train Set with SVM with *Radial* Kernel

svm\_bi1 = svm(default ~ .,  
 data = train\_bi,  
 kernel = 'radial',  
 gamma = 1,  
 cost = 1)  
  
summary(svm\_bi1)

##   
## Call:  
## svm(formula = default ~ ., data = train\_bi, kernel = "radial",   
## gamma = 1, cost = 1)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1   
## gamma: 1   
##   
## Number of Support Vectors: 671  
##   
## ( 435 236 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Tuning of SVM kernel

set.seed(111)  
  
tune\_out\_bi1 = tune(svm,  
 default ~ .,  
 data = train\_bi,  
 kernel = 'radial',  
 ranges = list(cost = c(0.1, 1, 10, 100, 1000),  
 gamma = c(0.5, 1, 2, 3, 4)),  
 tunecontrol = tune.control(sampling = 'cross',  
 cross = 5))  
  
summary(tune\_out\_bi1)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 5-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 100 0.5  
##   
## - best performance: 0.0265   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 1e-01 0.5 0.032875 0.006259368  
## 2 1e+00 0.5 0.027750 0.005548085  
## 3 1e+01 0.5 0.027125 0.005221650  
## 4 1e+02 0.5 0.026500 0.004605533  
## 5 1e+03 0.5 0.026750 0.005142501  
## 6 1e-01 1.0 0.032375 0.005473659  
## 7 1e+00 1.0 0.027000 0.004968652  
## 8 1e+01 1.0 0.027125 0.005350964  
## 9 1e+02 1.0 0.027000 0.005902859  
## 10 1e+03 1.0 0.028375 0.007023955  
## 11 1e-01 2.0 0.032500 0.005812836  
## 12 1e+00 2.0 0.027375 0.005785893  
## 13 1e+01 2.0 0.027250 0.005369183  
## 14 1e+02 2.0 0.028875 0.007672496  
## 15 1e+03 2.0 0.029625 0.008156190  
## 16 1e-01 3.0 0.033000 0.006486163  
## 17 1e+00 3.0 0.027625 0.005886292  
## 18 1e+01 3.0 0.028125 0.007043392  
## 19 1e+02 3.0 0.030125 0.008494024  
## 20 1e+03 3.0 0.030625 0.008489424  
## 21 1e-01 4.0 0.033500 0.006608470  
## 22 1e+00 4.0 0.027875 0.006101357  
## 23 1e+01 4.0 0.028875 0.007898279  
## 24 1e+02 4.0 0.031000 0.008599146  
## 25 1e+03 4.0 0.030000 0.007806247

After tuning, we discover that the best parameters are cost = 100 and gamma = 0.5.

bestmod\_bi1 = tune\_out\_bi1$best.model  
summary(bestmod\_bi1)

##   
## Call:  
## best.tune(method = svm, train.x = default ~ ., data = train\_bi,   
## ranges = list(cost = c(0.1, 1, 10, 100, 1000), gamma = c(0.5,   
## 1, 2, 3, 4)), tunecontrol = tune.control(sampling = "cross",   
## cross = 5), kernel = "radial")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 100   
## gamma: 0.5   
##   
## Number of Support Vectors: 540  
##   
## ( 314 226 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Confusion Matrix

# CM on Test Set  
newpred\_bi\_test1 = predict(bestmod\_bi1, test\_bi)  
confusionMatrix(table(prediction = newpred\_bi\_test1,  
 actual = test\_bi$default))

## Confusion Matrix and Statistics  
##   
## actual  
## prediction No Yes  
## No 1928 53  
## Yes 5 14  
##   
## Accuracy : 0.971   
## 95% CI : (0.9627, 0.9779)  
## No Information Rate : 0.9665   
## P-Value [Acc > NIR] : 0.1447   
##   
## Kappa : 0.3154   
## Mcnemar's Test P-Value : 6.769e-10   
##   
## Sensitivity : 0.9974   
## Specificity : 0.2090   
## Pos Pred Value : 0.9732   
## Neg Pred Value : 0.7368   
## Prevalence : 0.9665   
## Detection Rate : 0.9640   
## Detection Prevalence : 0.9905   
## Balanced Accuracy : 0.6032   
##   
## 'Positive' Class : No   
##

**97.1%** accuracy on test set achieved.

### Fitting Train Set with SVM with *Sigmoid* Kernel

svm\_bi2 = svm(default ~ .,  
 data = train\_bi,  
 kernel = 'sigmoid',  
 gamma = 1,  
 cost = 1)  
  
summary(svm\_bi2)

##   
## Call:  
## svm(formula = default ~ ., data = train\_bi, kernel = "sigmoid",   
## gamma = 1, cost = 1)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: sigmoid   
## cost: 1   
## gamma: 1   
## coef.0: 0   
##   
## Number of Support Vectors: 501  
##   
## ( 251 250 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Tuning of SVM Kernel

set.seed(111)  
  
tune\_out\_bi2 = tune(svm,  
 default ~ .,  
 data = train\_bi,  
 kernel = 'sigmoid',  
 ranges = list(cost = c(0.1, 1, 10, 100, 1000),  
 gamma = c(0.5, 1, 2, 3, 4)),  
 tunecontrol = tune.control(sampling = 'cross',  
 cross = 5))  
  
summary(tune\_out\_bi2)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 5-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 0.1 0.5  
##   
## - best performance: 0.042875   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 1e-01 0.5 0.042875 0.005495026  
## 2 1e+00 0.5 0.052625 0.006379900  
## 3 1e+01 0.5 0.053875 0.007199718  
## 4 1e+02 0.5 0.053875 0.007062777  
## 5 1e+03 0.5 0.054000 0.007323443  
## 6 1e-01 1.0 0.045375 0.006519202  
## 7 1e+00 1.0 0.057875 0.008071942  
## 8 1e+01 1.0 0.060250 0.006274950  
## 9 1e+02 1.0 0.060250 0.006274950  
## 10 1e+03 1.0 0.060250 0.006274950  
## 11 1e-01 2.0 0.044500 0.007426725  
## 12 1e+00 2.0 0.060500 0.008141810  
## 13 1e+01 2.0 0.062375 0.007569986  
## 14 1e+02 2.0 0.063000 0.007710585  
## 15 1e+03 2.0 0.063125 0.007868549  
## 16 1e-01 3.0 0.045375 0.006855085  
## 17 1e+00 3.0 0.058750 0.006774839  
## 18 1e+01 3.0 0.061375 0.006937218  
## 19 1e+02 3.0 0.061375 0.006780602  
## 20 1e+03 3.0 0.061375 0.006780602  
## 21 1e-01 4.0 0.044375 0.005519851  
## 22 1e+00 4.0 0.060875 0.007283329  
## 23 1e+01 4.0 0.061375 0.006425657  
## 24 1e+02 4.0 0.061625 0.006459005  
## 25 1e+03 4.0 0.061625 0.006459005

After tuning, the best parameters are cost = 0.1 and gamma = 0.5.

bestmod\_bi2 = tune\_out\_bi2$best.model  
summary(bestmod\_bi2)

##   
## Call:  
## best.tune(method = svm, train.x = default ~ ., data = train\_bi,   
## ranges = list(cost = c(0.1, 1, 10, 100, 1000), gamma = c(0.5,   
## 1, 2, 3, 4)), tunecontrol = tune.control(sampling = "cross",   
## cross = 5), kernel = "sigmoid")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: sigmoid   
## cost: 0.1   
## gamma: 0.5   
## coef.0: 0   
##   
## Number of Support Vectors: 521  
##   
## ( 261 260 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Confusion Matrix

# CM on Test Set  
newpred\_bi\_test2 = predict(bestmod\_bi2, test\_bi)  
confusionMatrix(table(prediction = newpred\_bi\_test2,  
 actual = test\_bi$default))

## Confusion Matrix and Statistics  
##   
## actual  
## prediction No Yes  
## No 1892 63  
## Yes 41 4  
##   
## Accuracy : 0.948   
## 95% CI : (0.9373, 0.9573)  
## No Information Rate : 0.9665   
## P-Value [Acc > NIR] : 0.99999   
##   
## Kappa : 0.0457   
## Mcnemar's Test P-Value : 0.03947   
##   
## Sensitivity : 0.97879   
## Specificity : 0.05970   
## Pos Pred Value : 0.96777   
## Neg Pred Value : 0.08889   
## Prevalence : 0.96650   
## Detection Rate : 0.94600   
## Detection Prevalence : 0.97750   
## Balanced Accuracy : 0.51925   
##   
## 'Positive' Class : No   
##

Accuracy of **94.8%** achieved on test set.

## Balance & Student Dataset (BS)

### Fitting Train Set with SVM with *Radial* Kernel

svm\_bs1 = svm(default ~ .,  
 data = train\_bs,  
 kernel = 'radial',  
 gamma = 1,  
 cost = 1)  
  
summary(svm\_bs1)

##   
## Call:  
## svm(formula = default ~ ., data = train\_bs, kernel = "radial",   
## gamma = 1, cost = 1)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1   
## gamma: 1   
##   
## Number of Support Vectors: 577  
##   
## ( 343 234 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Tuning of SVM Kernel

set.seed(111)  
  
tune\_out\_bs1 = tune(svm,  
 default ~ .,  
 data = train\_bs,  
 kernel = 'radial',  
 ranges = list(cost = c(0.1, 1, 10, 100, 1000),  
 gamma = c(0.5, 1, 2, 3, 4)),  
 tunecontrol = tune.control(sampling = 'cross',  
 cross = 5))  
  
summary(tune\_out\_bs1)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 5-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 1000 1  
##   
## - best performance: 0.026875   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 1e-01 0.5 0.031750 0.005785893  
## 2 1e+00 0.5 0.027000 0.005491471  
## 3 1e+01 0.5 0.027250 0.005240318  
## 4 1e+02 0.5 0.027250 0.004812906  
## 5 1e+03 0.5 0.027500 0.005000000  
## 6 1e-01 1.0 0.029000 0.005704302  
## 7 1e+00 1.0 0.027250 0.005240318  
## 8 1e+01 1.0 0.027250 0.005240318  
## 9 1e+02 1.0 0.027250 0.005512769  
## 10 1e+03 1.0 0.026875 0.005711146  
## 11 1e-01 2.0 0.028500 0.005277458  
## 12 1e+00 2.0 0.027125 0.005477226  
## 13 1e+01 2.0 0.027125 0.005906167  
## 14 1e+02 2.0 0.027375 0.005968668  
## 15 1e+03 2.0 0.028125 0.006903351  
## 16 1e-01 3.0 0.028625 0.004968652  
## 17 1e+00 3.0 0.027125 0.005906167  
## 18 1e+01 3.0 0.027375 0.005968668  
## 19 1e+02 3.0 0.028000 0.006649718  
## 20 1e+03 3.0 0.027750 0.006459005  
## 21 1e-01 4.0 0.029250 0.005255206  
## 22 1e+00 4.0 0.027375 0.005968668  
## 23 1e+01 4.0 0.027625 0.005902859  
## 24 1e+02 4.0 0.028000 0.006965316  
## 25 1e+03 4.0 0.027750 0.007376589

Best parameters are cost = 1000 and gamma = 1.

bestmod\_bs1 = tune\_out\_bs1$best.model  
summary(bestmod\_bs1)

##   
## Call:  
## best.tune(method = svm, train.x = default ~ ., data = train\_bs,   
## ranges = list(cost = c(0.1, 1, 10, 100, 1000), gamma = c(0.5,   
## 1, 2, 3, 4)), tunecontrol = tune.control(sampling = "cross",   
## cross = 5), kernel = "radial")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1000   
## gamma: 1   
##   
## Number of Support Vectors: 480  
##   
## ( 256 224 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Confusion Matrix

# CM on Test Set  
newpred\_bs\_test1 = predict(bestmod\_bs1, test\_bs)  
confusionMatrix(table(prediction = newpred\_bs\_test1,  
 actual = test\_bs$default))

## Confusion Matrix and Statistics  
##   
## actual  
## prediction No Yes  
## No 1928 52  
## Yes 5 15  
##   
## Accuracy : 0.9715   
## 95% CI : (0.9632, 0.9783)  
## No Information Rate : 0.9665   
## P-Value [Acc > NIR] : 0.1172   
##   
## Kappa : 0.3346   
## Mcnemar's Test P-Value : 1.109e-09   
##   
## Sensitivity : 0.9974   
## Specificity : 0.2239   
## Pos Pred Value : 0.9737   
## Neg Pred Value : 0.7500   
## Prevalence : 0.9665   
## Detection Rate : 0.9640   
## Detection Prevalence : 0.9900   
## Balanced Accuracy : 0.6106   
##   
## 'Positive' Class : No   
##

**97.2%** achieved on test set.

### Fitting Train Set with SVM with *Sigmoid* Kernel

svm\_bs2 = svm(default ~ .,  
 data = train\_bs,  
 kernel = 'sigmoid',  
 gamma = 1,  
 cost = 1)  
  
summary(svm\_bs2)

##   
## Call:  
## svm(formula = default ~ ., data = train\_bs, kernel = "sigmoid",   
## gamma = 1, cost = 1)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: sigmoid   
## cost: 1   
## gamma: 1   
## coef.0: 0   
##   
## Number of Support Vectors: 371  
##   
## ( 186 185 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Tuning of SVM Kernel

set.seed(111)  
  
tune\_out\_bs2 = tune(svm,  
 default ~ .,  
 data = train\_bs,  
 kernel = 'sigmoid',  
 ranges = list(cost = c(0.1, 1, 10, 100, 1000),  
 gamma = c(0.5, 1, 2, 3, 4)),  
 tunecontrol = tune.control(sampling = 'cross',  
 cross = 5))  
  
summary(tune\_out\_bs2)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 5-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 0.1 2  
##   
## - best performance: 0.0325   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 1e-01 0.5 0.033250 0.006364575  
## 2 1e+00 0.5 0.037750 0.007863583  
## 3 1e+01 0.5 0.041625 0.005789268  
## 4 1e+02 0.5 0.046875 0.008160978  
## 5 1e+03 0.5 0.050625 0.007967218  
## 6 1e-01 1.0 0.033250 0.006364575  
## 7 1e+00 1.0 0.033750 0.004571481  
## 8 1e+01 1.0 0.051125 0.007621413  
## 9 1e+02 1.0 0.061375 0.009136551  
## 10 1e+03 1.0 0.060500 0.010801837  
## 11 1e-01 2.0 0.032500 0.006234355  
## 12 1e+00 2.0 0.034125 0.004321097  
## 13 1e+01 2.0 0.044000 0.003660388  
## 14 1e+02 2.0 0.044125 0.003765593  
## 15 1e+03 2.0 0.044250 0.003888083  
## 16 1e-01 3.0 0.033250 0.006364575  
## 17 1e+00 3.0 0.038875 0.004293891  
## 18 1e+01 3.0 0.049875 0.012834585  
## 19 1e+02 3.0 0.050750 0.014013108  
## 20 1e+03 3.0 0.050875 0.013972629  
## 21 1e-01 4.0 0.033250 0.006364575  
## 22 1e+00 4.0 0.039000 0.006413487  
## 23 1e+01 4.0 0.050500 0.012925568  
## 24 1e+02 4.0 0.050875 0.012644478  
## 25 1e+03 4.0 0.050875 0.012644478

Best parameters are cost = 0.1 and gamma = 2.

bestmod\_bs2 = tune\_out\_bs2$best.model  
summary(bestmod\_bs2)

##   
## Call:  
## best.tune(method = svm, train.x = default ~ ., data = train\_bs,   
## ranges = list(cost = c(0.1, 1, 10, 100, 1000), gamma = c(0.5,   
## 1, 2, 3, 4)), tunecontrol = tune.control(sampling = "cross",   
## cross = 5), kernel = "sigmoid")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: sigmoid   
## cost: 0.1   
## gamma: 2   
## coef.0: 0   
##   
## Number of Support Vectors: 533  
##   
## ( 267 266 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Confusion Matrix

# CM on Test Set  
newpred\_bs\_test2 = predict(bestmod\_bs2, test\_bs)  
confusionMatrix(table(prediction = newpred\_bs\_test2,  
 actual = test\_bs$default))

## Confusion Matrix and Statistics  
##   
## actual  
## prediction No Yes  
## No 1932 63  
## Yes 1 4  
##   
## Accuracy : 0.968   
## 95% CI : (0.9593, 0.9753)  
## No Information Rate : 0.9665   
## P-Value [Acc > NIR] : 0.3847   
##   
## Kappa : 0.107   
## Mcnemar's Test P-Value : 2.44e-14   
##   
## Sensitivity : 0.9995   
## Specificity : 0.0597   
## Pos Pred Value : 0.9684   
## Neg Pred Value : 0.8000   
## Prevalence : 0.9665   
## Detection Rate : 0.9660   
## Detection Prevalence : 0.9975   
## Balanced Accuracy : 0.5296   
##   
## 'Positive' Class : No   
##

**96.8%** achieved on test set.

## Income & Student Dataset (IS)

### Fitting Train Set with SVM with *Radial* Kernel

svm\_is1 = svm(default ~ .,  
 data = train\_is,  
 kernel = 'radial',  
 gamma = 1,  
 cost = 1)  
  
summary(svm\_is1)

##   
## Call:  
## svm(formula = default ~ ., data = train\_is, kernel = "radial",   
## gamma = 1, cost = 1)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1   
## gamma: 1   
##   
## Number of Support Vectors: 612  
##   
## ( 346 266 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Tuning of SVM Kernel

set.seed(111)  
  
tune\_out\_is1 = tune(svm,  
 default ~ .,  
 data = train\_is,  
 kernel = 'radial',  
 ranges = list(cost = c(0.1, 1, 10, 100, 1000),  
 gamma = c(0.5, 1, 2, 3, 4)),  
 tunecontrol = tune.control(sampling = 'cross',  
 cross = 5))  
  
summary(tune\_out\_is1)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 5-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 0.1 0.5  
##   
## - best performance: 0.03325   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 1e-01 0.5 0.03325 0.006364575  
## 2 1e+00 0.5 0.03325 0.006364575  
## 3 1e+01 0.5 0.03325 0.006364575  
## 4 1e+02 0.5 0.03325 0.006364575  
## 5 1e+03 0.5 0.03325 0.006364575  
## 6 1e-01 1.0 0.03325 0.006364575  
## 7 1e+00 1.0 0.03325 0.006364575  
## 8 1e+01 1.0 0.03325 0.006364575  
## 9 1e+02 1.0 0.03325 0.006364575  
## 10 1e+03 1.0 0.03325 0.006364575  
## 11 1e-01 2.0 0.03325 0.006364575  
## 12 1e+00 2.0 0.03325 0.006364575  
## 13 1e+01 2.0 0.03325 0.006364575  
## 14 1e+02 2.0 0.03325 0.006364575  
## 15 1e+03 2.0 0.03325 0.006364575  
## 16 1e-01 3.0 0.03325 0.006364575  
## 17 1e+00 3.0 0.03325 0.006364575  
## 18 1e+01 3.0 0.03325 0.006364575  
## 19 1e+02 3.0 0.03325 0.006364575  
## 20 1e+03 3.0 0.03325 0.006364575  
## 21 1e-01 4.0 0.03325 0.006364575  
## 22 1e+00 4.0 0.03325 0.006364575  
## 23 1e+01 4.0 0.03325 0.006364575  
## 24 1e+02 4.0 0.03325 0.006364575  
## 25 1e+03 4.0 0.03325 0.006364575

Best parameters are cost = 0.1 and gamma = 0.5.

bestmod\_is1 = tune\_out\_is1$best.model  
summary(bestmod\_is1)

##   
## Call:  
## best.tune(method = svm, train.x = default ~ ., data = train\_is,   
## ranges = list(cost = c(0.1, 1, 10, 100, 1000), gamma = c(0.5,   
## 1, 2, 3, 4)), tunecontrol = tune.control(sampling = "cross",   
## cross = 5), kernel = "radial")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 0.1   
## gamma: 0.5   
##   
## Number of Support Vectors: 553  
##   
## ( 287 266 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Confusion Matrix

# CM on Test Set  
newpred\_is\_test1 = predict(bestmod\_is1, test\_is)  
confusionMatrix(table(prediction = newpred\_is\_test1,  
 actual = test\_is$default))

## Confusion Matrix and Statistics  
##   
## actual  
## prediction No Yes  
## No 1933 67  
## Yes 0 0  
##   
## Accuracy : 0.9665   
## 95% CI : (0.9576, 0.9739)  
## No Information Rate : 0.9665   
## P-Value [Acc > NIR] : 0.5324   
##   
## Kappa : 0   
## Mcnemar's Test P-Value : 7.433e-16   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.9665   
## Neg Pred Value : NaN   
## Prevalence : 0.9665   
## Detection Rate : 0.9665   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : No   
##

Accuracy of **96.7%** achieved on test set.

### Fitting Train Set with SVM with *Sigmoid* Kernel

svm\_is2 = svm(default ~ .,  
 data = train\_is,  
 kernel = 'sigmoid',  
 gamma = 1,  
 cost = 1)  
  
summary(svm\_is2)

##   
## Call:  
## svm(formula = default ~ ., data = train\_is, kernel = "sigmoid",   
## gamma = 1, cost = 1)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: sigmoid   
## cost: 1   
## gamma: 1   
## coef.0: 0   
##   
## Number of Support Vectors: 533  
##   
## ( 267 266 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Tuning of SVM Kernel

set.seed(111)  
  
tune\_out\_is2 = tune(svm,  
 default ~ .,  
 data = train\_is,  
 kernel = 'sigmoid',  
 ranges = list(cost = c(0.1, 1, 10, 100, 1000),  
 gamma = c(0.5, 1, 2, 3, 4)),  
 tunecontrol = tune.control(sampling = 'cross',  
 cross = 5))  
  
summary(tune\_out\_is2)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 5-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 0.1 0.5  
##   
## - best performance: 0.036   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 1e-01 0.5 0.036000 0.005350964  
## 2 1e+00 0.5 0.056000 0.007662306  
## 3 1e+01 0.5 0.061875 0.008648970  
## 4 1e+02 0.5 0.065625 0.002724312  
## 5 1e+03 0.5 0.066750 0.002592055  
## 6 1e-01 1.0 0.036375 0.005919380  
## 7 1e+00 1.0 0.054000 0.013481759  
## 8 1e+01 1.0 0.056250 0.013506364  
## 9 1e+02 1.0 0.064125 0.005165238  
## 10 1e+03 1.0 0.064250 0.004909334  
## 11 1e-01 2.0 0.037375 0.007062777  
## 12 1e+00 2.0 0.055750 0.015215894  
## 13 1e+01 2.0 0.063500 0.011722708  
## 14 1e+02 2.0 0.067500 0.008232462  
## 15 1e+03 2.0 0.068000 0.007634216  
## 16 1e-01 3.0 0.037500 0.007084865  
## 17 1e+00 3.0 0.055875 0.015023419  
## 18 1e+01 3.0 0.065250 0.005441450  
## 19 1e+02 3.0 0.066750 0.003937996  
## 20 1e+03 3.0 0.066875 0.003671044  
## 21 1e-01 4.0 0.040625 0.008184876  
## 22 1e+00 4.0 0.056750 0.015893936  
## 23 1e+01 4.0 0.066500 0.006474107  
## 24 1e+02 4.0 0.068000 0.005579679  
## 25 1e+03 4.0 0.068000 0.005579679

Best parameters are cost = 0.1 and gamma = 0.5.

bestmod\_is2 = tune\_out\_is2$best.model  
summary(bestmod\_is2)

##   
## Call:  
## best.tune(method = svm, train.x = default ~ ., data = train\_is,   
## ranges = list(cost = c(0.1, 1, 10, 100, 1000), gamma = c(0.5,   
## 1, 2, 3, 4)), tunecontrol = tune.control(sampling = "cross",   
## cross = 5), kernel = "sigmoid")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: sigmoid   
## cost: 0.1   
## gamma: 0.5   
## coef.0: 0   
##   
## Number of Support Vectors: 534  
##   
## ( 268 266 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Confusion Matrix

# CM on Test Set  
newpred\_is\_test2 = predict(bestmod\_is2, test\_is)  
confusionMatrix(table(prediction = newpred\_is\_test2,  
 actual = test\_is$default))

## Confusion Matrix and Statistics  
##   
## actual  
## prediction No Yes  
## No 1932 67  
## Yes 1 0  
##   
## Accuracy : 0.966   
## 95% CI : (0.9571, 0.9735)  
## No Information Rate : 0.9665   
## P-Value [Acc > NIR] : 0.5812   
##   
## Kappa : -0.001   
## Mcnemar's Test P-Value : 3.211e-15   
##   
## Sensitivity : 0.9995   
## Specificity : 0.0000   
## Pos Pred Value : 0.9665   
## Neg Pred Value : 0.0000   
## Prevalence : 0.9665   
## Detection Rate : 0.9660   
## Detection Prevalence : 0.9995   
## Balanced Accuracy : 0.4997   
##   
## 'Positive' Class : No   
##

Accuracy of **96.7%** achieved on test set.

## Full Training Dataset (ALL)

### Fitting Train Set with SVM with *Radial* Kernel

Finally, we consider the dataset with *all* predictor variables.

svm\_all1 = svm(default ~ .,  
 data = train,  
 kernel = 'radial',  
 gamma = 1,  
 cost = 1)  
  
summary(svm\_all1)

##   
## Call:  
## svm(formula = default ~ ., data = train, kernel = "radial", gamma = 1,   
## cost = 1)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1   
## gamma: 1   
##   
## Number of Support Vectors: 691  
##   
## ( 453 238 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Tuning of SVM Kernel

set.seed(111)  
  
tune\_out\_all1 = tune(svm,  
 default ~ .,  
 data = train,  
 kernel = 'radial',  
 ranges = list(cost = c(0.1, 1, 10, 100, 1000),  
 gamma = c(0.5, 1, 2, 3, 4)),  
 tunecontrol = tune.control(sampling = 'cross',  
 cross = 5))  
  
summary(tune\_out\_all1)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 5-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 10 2  
##   
## - best performance: 0.026625   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 1e-01 0.5 0.033250 0.006364575  
## 2 1e+00 0.5 0.027250 0.005822907  
## 3 1e+01 0.5 0.027375 0.005123475  
## 4 1e+02 0.5 0.026750 0.004558646  
## 5 1e+03 0.5 0.026875 0.005625000  
## 6 1e-01 1.0 0.033000 0.006256247  
## 7 1e+00 1.0 0.027250 0.005240318  
## 8 1e+01 1.0 0.027250 0.005295930  
## 9 1e+02 1.0 0.027125 0.006149187  
## 10 1e+03 1.0 0.028250 0.007400380  
## 11 1e-01 2.0 0.033125 0.006327643  
## 12 1e+00 2.0 0.027375 0.005785893  
## 13 1e+01 2.0 0.026625 0.005806112  
## 14 1e+02 2.0 0.028750 0.007447735  
## 15 1e+03 2.0 0.030750 0.007531185  
## 16 1e-01 3.0 0.033375 0.006826534  
## 17 1e+00 3.0 0.027875 0.006101357  
## 18 1e+01 3.0 0.028000 0.007090376  
## 19 1e+02 3.0 0.030500 0.008354920  
## 20 1e+03 3.0 0.031250 0.008291562  
## 21 1e-01 4.0 0.033375 0.006652655  
## 22 1e+00 4.0 0.028000 0.006066043  
## 23 1e+01 4.0 0.029500 0.007582875  
## 24 1e+02 4.0 0.030875 0.008484821  
## 25 1e+03 4.0 0.030500 0.008033135

Best parameters are cost = 10 and gamma = 2.

bestmod\_all1 = tune\_out\_all1$best.model  
summary(bestmod\_all1)

##   
## Call:  
## best.tune(method = svm, train.x = default ~ ., data = train,   
## ranges = list(cost = c(0.1, 1, 10, 100, 1000), gamma = c(0.5,   
## 1, 2, 3, 4)), tunecontrol = tune.control(sampling = "cross",   
## cross = 5), kernel = "radial")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 10   
## gamma: 2   
##   
## Number of Support Vectors: 741  
##   
## ( 513 228 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Confusion Matrix

# CM on Test Set  
newpred\_all\_test1 = predict(bestmod\_all1, test)  
confusionMatrix(table(prediction = newpred\_all\_test1,  
 actual = test$default))

## Confusion Matrix and Statistics  
##   
## actual  
## prediction No Yes  
## No 1928 52  
## Yes 5 15  
##   
## Accuracy : 0.9715   
## 95% CI : (0.9632, 0.9783)  
## No Information Rate : 0.9665   
## P-Value [Acc > NIR] : 0.1172   
##   
## Kappa : 0.3346   
## Mcnemar's Test P-Value : 1.109e-09   
##   
## Sensitivity : 0.9974   
## Specificity : 0.2239   
## Pos Pred Value : 0.9737   
## Neg Pred Value : 0.7500   
## Prevalence : 0.9665   
## Detection Rate : 0.9640   
## Detection Prevalence : 0.9900   
## Balanced Accuracy : 0.6106   
##   
## 'Positive' Class : No   
##

Accuracy of **97.2%** achieved on test set.

### Fitting Train Set with SVM with *Sigmoid* Kernel

svm\_all2 = svm(default ~ .,  
 data = train,  
 kernel = 'sigmoid',  
 gamma = 1,  
 cost = 1)  
  
summary(svm\_all2)

##   
## Call:  
## svm(formula = default ~ ., data = train, kernel = "sigmoid",   
## gamma = 1, cost = 1)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: sigmoid   
## cost: 1   
## gamma: 1   
## coef.0: 0   
##   
## Number of Support Vectors: 500  
##   
## ( 250 250 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Tuning of SVM Kernel

set.seed(111)  
  
tune\_out\_all2 = tune(svm,  
 default ~ .,  
 data = train,  
 kernel = 'sigmoid',  
 ranges = list(cost = c(0.1, 1, 10, 100, 1000),  
 gamma = c(0.5, 1, 2, 3, 4)),  
 tunecontrol = tune.control(sampling = 'cross',  
 cross = 5))  
  
summary(tune\_out\_all2)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 5-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 0.1 0.5  
##   
## - best performance: 0.041875   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 1e-01 0.5 0.041875 0.005779138  
## 2 1e+00 0.5 0.048750 0.005096721  
## 3 1e+01 0.5 0.050125 0.004406139  
## 4 1e+02 0.5 0.050250 0.004298437  
## 5 1e+03 0.5 0.050250 0.004298437  
## 6 1e-01 1.0 0.044750 0.005441450  
## 7 1e+00 1.0 0.058625 0.008642193  
## 8 1e+01 1.0 0.060750 0.008608226  
## 9 1e+02 1.0 0.060750 0.008459462  
## 10 1e+03 1.0 0.060000 0.006903351  
## 11 1e-01 2.0 0.044000 0.006608470  
## 12 1e+00 2.0 0.060250 0.007161638  
## 13 1e+01 2.0 0.062000 0.007710585  
## 14 1e+02 2.0 0.062250 0.007826238  
## 15 1e+03 2.0 0.062375 0.007848666  
## 16 1e-01 3.0 0.045000 0.006945660  
## 17 1e+00 3.0 0.060250 0.006290494  
## 18 1e+01 3.0 0.062000 0.005199159  
## 19 1e+02 3.0 0.062000 0.005199159  
## 20 1e+03 3.0 0.062125 0.005314338  
## 21 1e-01 4.0 0.044625 0.005922679  
## 22 1e+00 4.0 0.061000 0.007023955  
## 23 1e+01 4.0 0.063125 0.007167091  
## 24 1e+02 4.0 0.063375 0.007079349  
## 25 1e+03 4.0 0.063375 0.007079349

Best parameters are cost = 0.1 and gamma = 0.5.

bestmod\_all2 = tune\_out\_all2$best.model  
summary(bestmod\_all2)

##   
## Call:  
## best.tune(method = svm, train.x = default ~ ., data = train,   
## ranges = list(cost = c(0.1, 1, 10, 100, 1000), gamma = c(0.5,   
## 1, 2, 3, 4)), tunecontrol = tune.control(sampling = "cross",   
## cross = 5), kernel = "sigmoid")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: sigmoid   
## cost: 0.1   
## gamma: 0.5   
## coef.0: 0   
##   
## Number of Support Vectors: 520  
##   
## ( 260 260 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

### Confusion Matrix

# CM on Test Set  
newpred\_all\_test2 = predict(bestmod\_all2, test)  
confusionMatrix(table(prediction = newpred\_all\_test2,  
 actual = test$default))

## Confusion Matrix and Statistics  
##   
## actual  
## prediction No Yes  
## No 1899 62  
## Yes 34 5  
##   
## Accuracy : 0.952   
## 95% CI : (0.9417, 0.9609)  
## No Information Rate : 0.9665   
## P-Value [Acc > NIR] : 0.999733   
##   
## Kappa : 0.0714   
## Mcnemar's Test P-Value : 0.005857   
##   
## Sensitivity : 0.98241   
## Specificity : 0.07463   
## Pos Pred Value : 0.96838   
## Neg Pred Value : 0.12821   
## Prevalence : 0.96650   
## Detection Rate : 0.94950   
## Detection Prevalence : 0.98050   
## Balanced Accuracy : 0.52852   
##   
## 'Positive' Class : No   
##

Accuracy of **95.2%** achieved on test set.

## Conclusion

After building the SVM models for all combinations of inputs, we are able to make some discoveries about the models used.

### Comparison of Best Parameters

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Control  Params | Radial Basis Function Kernel | | | | Sigmoid Kernel | | | |
|  | **BI** | **BS** | **IS** | **ALL** | **BI** | **BS** | **IS** | **ALL** |
| **Cost** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **Gamma** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **No. of SVs** | 671 | 577 | 612 | 691 | 501 | 371 | 533 | 500 |
| **Best Params** | **Radial Basis Function Kernel** | | | | **Sigmoid Kernel** | | | |
|  | **BI** | **BS** | **IS** | **ALL** | **BI** | **BS** | **IS** | **ALL** |
| **Cost** | 100 | 1000 | 0.1 | 10 | 0.1 | 0.2 | 0.1 | 0.1 |
| **Gamma** | 0.5 | 1 | 0.5 | 2 | 0.5 | 2 | 0.5 | 0.5 |
| **No. of SVs** | 540 | 480 | 553 | 741 | 521 | 533 | 534 | 520 |

### Test Set Accuracy Rates

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Radial Basis Function Kernel | | | | Sigmoid Kernel | | | |
| **BI** | **BS** | **IS** | **ALL** | **BI** | **BS** | **IS** | **ALL** |
| 97.1% | 97.2% | 96.7% | 97.2% | 94.8% | 96.8% | 96.7% | 95.2% |

We can infer from the results that the RBF kernel is generally superior in predicting the test set, as compared to the sigmoid kernel. However, there is one case whereby there is no difference in the accuracy rates between the two kernels, for the IS dataset.

Focusing on the BS dataset with RBF kernel where the highest accuracy rates were obtained, it would appear that when the cost was increased and the gamma held constant at 1, we get a narrower margin (fewer SVMs) and hence a tighter fit to the data. This would explain why we obtain a good accuracy in this case. We contrast this with the case of the sigmoid kernel where cost was much lower at 0.1 and gamma was higher at 2, this would result in a wider margin (more SVs). Misclassification error tends to be higher in this case with a wider margin.

It is also useful to note that including all variables for training may not necessarily improve our model. Thu number of SVs are a lot higher which leads to more bias and less robustness of the SVM.