## KE5206 | Computational Intelligence I CA2

## 

## 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')]

In addition, we have created subsets of the training and test sets by only selecting 2 variables each. One subset will have *balance* and *income* as the predictors (BI dataset), while the other subset will have *balance* and *student* (BS dataset).

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

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

We first work on the dataset with the *balance* and *income* variables. We are ignoring the *student* variable in this case.

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)))  
  
summary(tune\_out\_bi1)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 100 0.5  
##   
## - best performance: 0.026875   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 1e-01 0.5 0.032875 0.006589396  
## 2 1e+00 0.5 0.028000 0.005957022  
## 3 1e+01 0.5 0.027250 0.005916080  
## 4 1e+02 0.5 0.026875 0.005408648  
## 5 1e+03 0.5 0.027000 0.005658082  
## 6 1e-01 1.0 0.031000 0.005676462  
## 7 1e+00 1.0 0.027000 0.005868939  
## 8 1e+01 1.0 0.026875 0.005690208  
## 9 1e+02 1.0 0.027250 0.005945353  
## 10 1e+03 1.0 0.028000 0.006566963  
## 11 1e-01 2.0 0.031750 0.006043821  
## 12 1e+00 2.0 0.027125 0.005775006  
## 13 1e+01 2.0 0.027500 0.006454972  
## 14 1e+02 2.0 0.028250 0.006487167  
## 15 1e+03 2.0 0.028875 0.007008180  
## 16 1e-01 3.0 0.032125 0.006429371  
## 17 1e+00 3.0 0.027750 0.006146363  
## 18 1e+01 3.0 0.027875 0.005952649  
## 19 1e+02 3.0 0.028375 0.006873106  
## 20 1e+03 3.0 0.029125 0.007477977  
## 21 1e-01 4.0 0.032875 0.006847800  
## 22 1e+00 4.0 0.027750 0.006422616  
## 23 1e+01 4.0 0.027875 0.006456317  
## 24 1e+02 4.0 0.028750 0.006897061  
## 25 1e+03 4.0 0.029000 0.007066156

After tuning, we discover that the best parameters are cost = 100 and gamma = 1.

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)), 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

Number of support vectors have reduced in this case, when cost increased and gamma remains unchanged. This would imply that the margin has reduced.

### Confusion Matrices

# CM on Training Set  
newpred\_bi\_train1 = predict(bestmod\_bi1, train\_bi)  
confusionMatrix(table(prediction = newpred\_bi\_train1,  
 actual = train\_bi$default))

## Confusion Matrix and Statistics  
##   
## actual  
## prediction No Yes  
## No 7711 186  
## Yes 23 80  
##   
## Accuracy : 0.9739   
## 95% CI : (0.9701, 0.9773)  
## No Information Rate : 0.9668   
## P-Value [Acc > NIR] : 0.0001333   
##   
## Kappa : 0.4229   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9970   
## Specificity : 0.3008   
## Pos Pred Value : 0.9764   
## Neg Pred Value : 0.7767   
## Prevalence : 0.9667   
## Detection Rate : 0.9639   
## Detection Prevalence : 0.9871   
## Balanced Accuracy : 0.6489   
##   
## 'Positive' Class : No   
##

**97.4%** accuracy on training set achieved, which is pretty good.

# 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

We now build a SVM with a sigmoid kernel to determine if it is more effective than one with a radial basis 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)))  
  
summary(tune\_out\_bi2)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 0.1 0.5  
##   
## - best performance: 0.0445   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 1e-01 0.5 0.044500 0.007149204  
## 2 1e+00 0.5 0.053625 0.005415064  
## 3 1e+01 0.5 0.053250 0.003736085  
## 4 1e+02 0.5 0.052875 0.003586723  
## 5 1e+03 0.5 0.053000 0.003545341  
## 6 1e-01 1.0 0.046375 0.007871582  
## 7 1e+00 1.0 0.058625 0.008588793  
## 8 1e+01 1.0 0.060500 0.008783665  
## 9 1e+02 1.0 0.060500 0.008783665  
## 10 1e+03 1.0 0.060500 0.008783665  
## 11 1e-01 2.0 0.046625 0.007840644  
## 12 1e+00 2.0 0.059250 0.008583738  
## 13 1e+01 2.0 0.061250 0.008559433  
## 14 1e+02 2.0 0.061375 0.008608983  
## 15 1e+03 2.0 0.061375 0.008608983  
## 16 1e-01 3.0 0.046375 0.008608983  
## 17 1e+00 3.0 0.061625 0.008334375  
## 18 1e+01 3.0 0.062500 0.007095578  
## 19 1e+02 3.0 0.062750 0.007283924  
## 20 1e+03 3.0 0.062625 0.007155272  
## 21 1e-01 4.0 0.045250 0.008266398  
## 22 1e+00 4.0 0.061000 0.006739189  
## 23 1e+01 4.0 0.062750 0.007260051  
## 24 1e+02 4.0 0.063125 0.007174656  
## 25 1e+03 4.0 0.063125 0.007174656

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)), 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

Both cost and gamma parameters have been reduced after tuning, and number of support vectors has increased. There is now a larger margin due to a smaller cost and gamma.

### Confusion Matrices

# CM on Training Set  
newpred\_bi\_train2 = predict(bestmod\_bi2, train\_bi)  
confusionMatrix(table(prediction = newpred\_bi\_train2,  
 actual = train\_bi$default))

## Confusion Matrix and Statistics  
##   
## actual  
## prediction No Yes  
## No 7625 253  
## Yes 109 13  
##   
## Accuracy : 0.9548   
## 95% CI : (0.95, 0.9592)  
## No Information Rate : 0.9668   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.0471   
## Mcnemar's Test P-Value : 5.652e-14   
##   
## Sensitivity : 0.98591   
## Specificity : 0.04887   
## Pos Pred Value : 0.96789   
## Neg Pred Value : 0.10656   
## Prevalence : 0.96675   
## Detection Rate : 0.95312   
## Detection Prevalence : 0.98475   
## Balanced Accuracy : 0.51739   
##   
## 'Positive' Class : No   
##

Accuracy rate of **95.5%** achieved on training set.

# 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.

For this dataset, it is apparent that the SVM with a radial basis kernel has a better performance of 97.1% accuracy.

## Balance & Student Dataset

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

We now consider the dataset with only the *balance* and *student* variables.

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)))  
  
summary(tune\_out\_bs1)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 10 3  
##   
## - best performance: 0.026625   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 1e-01 0.5 0.030750 0.006324555  
## 2 1e+00 0.5 0.027125 0.006209592  
## 3 1e+01 0.5 0.027125 0.006096732  
## 4 1e+02 0.5 0.027375 0.005935124  
## 5 1e+03 0.5 0.027125 0.006068189  
## 6 1e-01 1.0 0.028625 0.005447030  
## 7 1e+00 1.0 0.027125 0.005894029  
## 8 1e+01 1.0 0.027125 0.005981743  
## 9 1e+02 1.0 0.027250 0.006003471  
## 10 1e+03 1.0 0.027125 0.005684103  
## 11 1e-01 2.0 0.027750 0.005614960  
## 12 1e+00 2.0 0.027375 0.005876330  
## 13 1e+01 2.0 0.027250 0.005945353  
## 14 1e+02 2.0 0.026750 0.006015027  
## 15 1e+03 2.0 0.026750 0.005898446  
## 16 1e-01 3.0 0.027625 0.005152197  
## 17 1e+00 3.0 0.027000 0.005957022  
## 18 1e+01 3.0 0.026625 0.005923412  
## 19 1e+02 3.0 0.026625 0.006039511  
## 20 1e+03 3.0 0.026875 0.006827487  
## 21 1e-01 4.0 0.028250 0.005407043  
## 22 1e+00 4.0 0.026750 0.006101002  
## 23 1e+01 4.0 0.026625 0.006039511  
## 24 1e+02 4.0 0.026875 0.006488505  
## 25 1e+03 4.0 0.027000 0.006406377

Best parameters are cost = 10 and gamma = 3.

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)), kernel = "radial")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 10   
## gamma: 3   
##   
## Number of Support Vectors: 837  
##   
## ( 615 222 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## No Yes

Number of support vectors have increased tremendously after using a moderate cost and high gamma. There is a much larger margin in this SVM.

### Confusion Matrices

# CM on Training Set  
newpred\_bs\_train1 = predict(bestmod\_bs1, train\_bs)  
confusionMatrix(table(prediction = newpred\_bs\_train1,  
 actual = train\_bs$default))

## Confusion Matrix and Statistics  
##   
## actual  
## prediction No Yes  
## No 7703 179  
## Yes 31 87  
##   
## Accuracy : 0.9738   
## 95% CI : (0.97, 0.9771)  
## No Information Rate : 0.9668   
## P-Value [Acc > NIR] : 0.0001727   
##   
## Kappa : 0.4417   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9960   
## Specificity : 0.3271   
## Pos Pred Value : 0.9773   
## Neg Pred Value : 0.7373   
## Prevalence : 0.9667   
## Detection Rate : 0.9629   
## Detection Prevalence : 0.9852   
## Balanced Accuracy : 0.6615   
##   
## 'Positive' Class : No   
##

Accuracy of **97.4%** achieved on training set.

# 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, which is very close to the performance of the radial basis kernel using the *balance* and *income* dataset.

### 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)))  
  
summary(tune\_out\_bs2)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 0.1 2  
##   
## - best performance: 0.03175   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 1e-01 0.5 0.033250 0.006851602  
## 2 1e+00 0.5 0.038625 0.008363487  
## 3 1e+01 0.5 0.039250 0.008744046  
## 4 1e+02 0.5 0.042250 0.009461178  
## 5 1e+03 0.5 0.048375 0.009447406  
## 6 1e-01 1.0 0.033250 0.006851602  
## 7 1e+00 1.0 0.034000 0.005797509  
## 8 1e+01 1.0 0.046875 0.011445068  
## 9 1e+02 1.0 0.048500 0.012173514  
## 10 1e+03 1.0 0.063750 0.008740074  
## 11 1e-01 2.0 0.031750 0.006213784  
## 12 1e+00 2.0 0.034250 0.006101002  
## 13 1e+01 2.0 0.042625 0.007179494  
## 14 1e+02 2.0 0.043000 0.006977145  
## 15 1e+03 2.0 0.043000 0.006977145  
## 16 1e-01 3.0 0.031875 0.006750772  
## 17 1e+00 3.0 0.036375 0.004267529  
## 18 1e+01 3.0 0.044625 0.005337563  
## 19 1e+02 3.0 0.045750 0.005502525  
## 20 1e+03 3.0 0.045750 0.005502525  
## 21 1e-01 4.0 0.033250 0.006851602  
## 22 1e+00 4.0 0.036375 0.006652328  
## 23 1e+01 4.0 0.045875 0.006796905  
## 24 1e+02 4.0 0.046375 0.007393926  
## 25 1e+03 4.0 0.046500 0.007402139

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)), 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

With cost reduced and gamma slightly increased, we now have more support vectors with a larger margin.

### Confusion Matrices

# CM on Training Set  
newpred\_bs\_train2 = predict(bestmod\_bs2, train\_bs)  
confusionMatrix(table(prediction = newpred\_bs\_train2,  
 actual = train\_bs$default))

## Confusion Matrix and Statistics  
##   
## actual  
## prediction No Yes  
## No 7728 240  
## Yes 6 26  
##   
## Accuracy : 0.9692   
## 95% CI : (0.9652, 0.9729)  
## No Information Rate : 0.9668   
## P-Value [Acc > NIR] : 0.111   
##   
## Kappa : 0.1686   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.99922   
## Specificity : 0.09774   
## Pos Pred Value : 0.96988   
## Neg Pred Value : 0.81250   
## Prevalence : 0.96675   
## Detection Rate : 0.96600   
## Detection Prevalence : 0.99600   
## Balanced Accuracy : 0.54848   
##   
## 'Positive' Class : No   
##

Accuracy of **96.9%** achieved on training set.

# 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.

The performance is not very different from that of the radial basis kernel, which suggests that using the *student* variable in analysis greatly helps to predict this binary classification problem.

## Conclusion

We compare the best parameters of the tuned models in the following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Best Params | BI (Radial) | BI (Sigmoid) | BS (Radial) | BS (Sigmoid) |
| Cost | 100 | 0.1 | **10** | 0.1 |
| Gamma | 1 | 0.5 | **3** | 2 |
| No. of SVs | 540 | 521 | **837** | 533 |

We are aware that the smaller the cost, the larger the margin and the more support vectors there will be. As for gamma, the higher it is, the more it allows the SVM to capture the shape of the data but there might be a risk of overfitting. There is also a large margin in our best model, which would explain the large number of support vectors.

We summarize the performances of the SVM kernels in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| BI (Radial) | BI (Sigmoid) | BS (Radial) | BS (Sigmoid) |
| 97.1% | 94.8% | **97.2%** | 96.8% |

The radial basis kernel has the better performance as compared to the sigmoid kernel. We also discovered that our accuracy rates are noticeably higher when using the BS dataset, which implies that the *student* variable is a more effective predictor than *income*.