Homwork 3

Hamed

2/18/2020

# Performance measurement, Imbalance correction and Imputing missing values

### 1. Load the Wisconsin breast cancer dataset from blackboard

## -- Attaching packages -------------------------------------------- tidyverse 1.3.0 --

## <U+2713> ggplot2 3.2.1 <U+2713> purrr 0.3.3  
## <U+2713> tibble 2.1.3 <U+2713> dplyr 0.8.3  
## <U+2713> tidyr 1.0.0 <U+2713> stringr 1.4.0  
## <U+2713> readr 1.3.1 <U+2713> forcats 0.4.0

## -- Conflicts ----------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

## Loading required package: rpart

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

## Loading required package: gplots

##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

## Loading required package: bitops

##   
## Attaching package: 'RCurl'

## The following object is masked from 'package:tidyr':  
##   
## complete

## Loaded gbm 2.1.5

## id\_number diagnosis radius\_mean texture\_mean perimeter\_mean area\_mean  
## 1 842302 M 17.99 10.38 122.80 1001.0  
## 2 842517 M 20.57 17.77 132.90 1326.0  
## 3 84300903 M 19.69 21.25 130.00 1203.0  
## 4 84348301 M 11.42 20.38 77.58 386.1  
## 5 84358402 M 20.29 14.34 135.10 1297.0  
## 6 843786 M 12.45 15.70 82.57 477.1  
## smoothness\_mean compactness\_mean concavity\_mean concave\_points\_mean  
## 1 0.11840 0.27760 0.3001 0.14710  
## 2 0.08474 0.07864 0.0869 0.07017  
## 3 0.10960 0.15990 0.1974 0.12790  
## 4 0.14250 0.28390 0.2414 0.10520  
## 5 0.10030 0.13280 0.1980 0.10430  
## 6 0.12780 0.17000 0.1578 0.08089  
## symmetry\_mean fractal\_dimension\_mean radius\_se texture\_se perimeter\_se  
## 1 0.2419 0.07871 1.0950 0.9053 8.589  
## 2 0.1812 0.05667 0.5435 0.7339 3.398  
## 3 0.2069 0.05999 0.7456 0.7869 4.585  
## 4 0.2597 0.09744 0.4956 1.1560 3.445  
## 5 0.1809 0.05883 0.7572 0.7813 5.438  
## 6 0.2087 0.07613 0.3345 0.8902 2.217  
## area\_se smoothness\_se compactness\_se concavity\_se concave\_points\_se  
## 1 153.40 0.006399 0.04904 0.05373 0.01587  
## 2 74.08 0.005225 0.01308 0.01860 0.01340  
## 3 94.03 0.006150 0.04006 0.03832 0.02058  
## 4 27.23 0.009110 0.07458 0.05661 0.01867  
## 5 94.44 0.011490 0.02461 0.05688 0.01885  
## 6 27.19 0.007510 0.03345 0.03672 0.01137  
## symmetry\_se fractal\_dimension\_se radius\_worst texture\_worst perimeter\_worst  
## 1 0.03003 0.006193 25.38 17.33 184.60  
## 2 0.01389 0.003532 24.99 23.41 158.80  
## 3 0.02250 0.004571 23.57 25.53 152.50  
## 4 0.05963 0.009208 14.91 26.50 98.87  
## 5 0.01756 0.005115 22.54 16.67 152.20  
## 6 0.02165 0.005082 15.47 23.75 103.40  
## area\_worst smoothness\_worst compactness\_worst concavity\_worst  
## 1 2019.0 0.1622 0.6656 0.7119  
## 2 1956.0 0.1238 0.1866 0.2416  
## 3 1709.0 0.1444 0.4245 0.4504  
## 4 567.7 0.2098 0.8663 0.6869  
## 5 1575.0 0.1374 0.2050 0.4000  
## 6 741.6 0.1791 0.5249 0.5355  
## concave\_points\_worst symmetry\_worst fractal\_dimension\_worst  
## 1 0.2654 0.4601 0.11890  
## 2 0.1860 0.2750 0.08902  
## 3 0.2430 0.3613 0.08758  
## 4 0.2575 0.6638 0.17300  
## 5 0.1625 0.2364 0.07678  
## 6 0.1741 0.3985 0.12440

### 2. How many missing data points are there?

sum(is.na(breast\_cancer)) #no missing values

## [1] 0

### In which columns are they missing?

which(is.na(breast\_cancer))

## integer(0)

### 3. Impute for the first column with missing values using the mean and round to an integer

breast\_cancer$radius\_mean[is.na(breast\_cancer$radius\_mean)]= round(mean(breast\_cancer$radius\_mean, na.rm=TRUE),digits = 0)

### 4. Impute for the second column with missing values using KNN with three nearest neighbors and round to integer

library(DMwR)

## Loading required package: grid

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

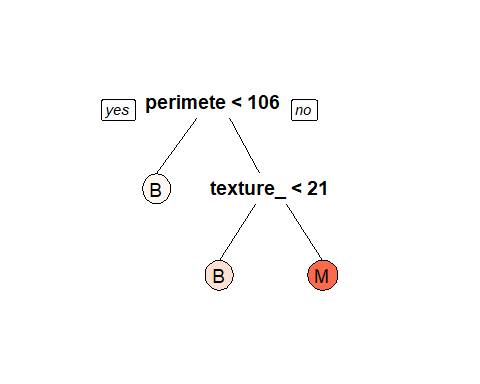
sec\_col=as.data.frame(breast\_cancer)  
breast\_cancer<-knnImputation(sec\_col,k=3,scale=T,meth="weighAvg",distData = NULL)  
dim(breast\_cancer) #There was no missing values

## [1] 569 32

### 5. Impute for the third column with missing values using a regression to predict the third column based on uniformity\_of\_cell\_shape, marginal\_adhesion and normal\_nucleoli. Round to integers.

### 6. Build a decision tree model to predict class using 80% training data and five fold cross validation with three repeats. Hint: Use method =" rpart" in caret

# Split the data into training and test set  
set.seed(123)  
training.samples <- breast\_cancer$diagnosis %>%   
 createDataPartition(p = 0.8, list = FALSE)  
train.data <- breast\_cancer[training.samples, ]  
test.data <- breast\_cancer[-training.samples, ]  
  
# Fit the model on the training set  
set.seed(123)  
model\_tree <- train(  
 diagnosis ~., data = train.data, method = "rpart",  
 trControl = trainControl("cv", number = 5),  
 tuneLength = 3  
)  
  
#visulize the decision tree  
prp(model\_tree$finalModel, box.palette = "Reds", tweak = 1.2)

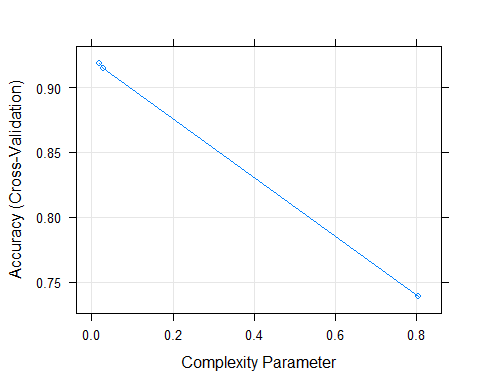


### 7. Show the accuracy results for each resample (fold) in the training data

model\_tree$results

## cp Accuracy Kappa AccuracySD KappaSD  
## 1 0.01764706 0.9188724 0.8272326 0.009676729 0.01958072  
## 2 0.02941176 0.9145246 0.8180973 0.015859232 0.03202774  
## 3 0.80588235 0.7386765 0.3202505 0.153817041 0.43857026

# Plot model accuracy vs different values of cp (complexity parameter)  
plot(model\_tree)



### 8. Do the following performance measures for the test dataset

* (a).Compute all the accuracy measures (Accuracy, sensitivity, specificity etc)

# Make predictions on the test data  
predicted.classes=model\_tree%>% predict(test.data)  
# Compute model accuracy rate on test data  
mean(predicted.classes == test.data$diagnosis)

## [1] 0.9026549

#calculate sensitivity  
  
sensitivity(predicted.classes,test.data$diagnosis)

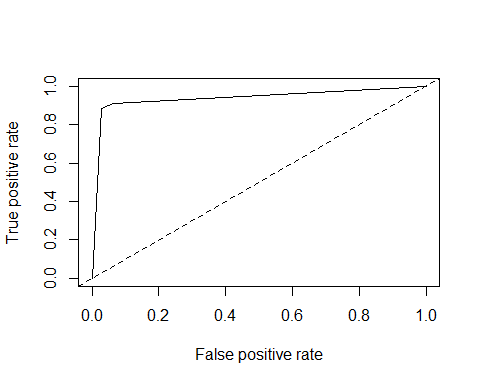
## [1] 0.9295775

#calculate specificity  
specificity(predicted.classes,test.data$diagnosis)

## [1] 0.8571429

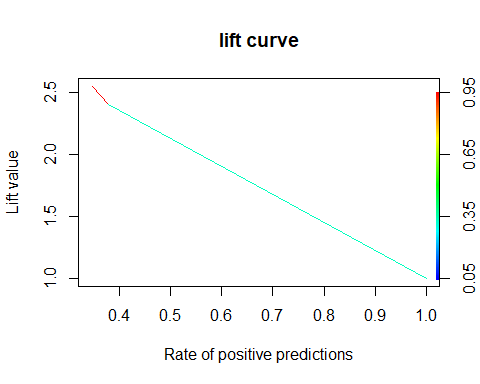
* (b).Plot the ROC curve

library(ROCR)  
pred <- prediction(predict(model\_tree, type = "prob")[, 2],train.data$diagnosis)  
plot(performance(pred, "tpr", "fpr"))  
abline(0, 1, lty = 2)



* (c).Plot the lift curve

perf <- performance(pred,"lift","rpp")  
plot(perf, main="lift curve", colorize=T)



* (d).Compute the AUC

auc = performance(pred, 'auc')  
slot(auc, 'y.values')

## [[1]]  
## [1] 0.9394282

### 9. Is the cancer data imbalanced by the class feature? What is the percentage of the majority class and the percentage of the minority class?

table(breast\_cancer$diagnosis) #calculates the class frequencies

##   
## B M   
## 357 212

tablature <- as.matrix(prop.table(table(breast\_cancer$diagnosis)) \* 100)  
tablature #gets the percentage: hence class "B" is the majority while class "M" is the minority

## [,1]  
## B 62.74165  
## M 37.25835

### 10. Now re-build the decision tree model above after correcting for imbalance using SMOTE.

# initialize imbalanced data  
imbal\_train=train.data  
imbal\_test=test.data  
  
# Set up control function for training  
ctrl <- trainControl(method = "repeatedcv",  
 number = 5,  
 repeats = 3,  
 classProbs = TRUE)

Build smote model

ctrl$sampling <- "smote"  
  
smote\_fit <- train(diagnosis ~ .,  
 data = imbal\_train,  
 method = "gbm",  
 verbose = FALSE,  
 metric = "ROC",  
 trControl = ctrl)  
smote\_fit

## Stochastic Gradient Boosting   
##   
## 456 samples  
## 31 predictor  
## 2 classes: 'B', 'M'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold, repeated 3 times)   
## Summary of sample sizes: 365, 364, 365, 365, 365, 365, ...   
## Addtional sampling using SMOTE  
##   
## Resampling results across tuning parameters:  
##   
## interaction.depth n.trees Accuracy Kappa   
## 1 50 0.9459229 0.8847245  
## 1 100 0.9583692 0.9113685  
## 1 150 0.9598105 0.9142149  
## 2 50 0.9590779 0.9126019  
## 2 100 0.9627090 0.9202105  
## 2 150 0.9649228 0.9249218  
## 3 50 0.9546584 0.9035574  
## 3 100 0.9605192 0.9155858  
## 3 150 0.9619764 0.9186721  
##   
## Tuning parameter 'shrinkage' was held constant at a value of 0.1  
##   
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were n.trees = 150, interaction.depth =  
## 2, shrinkage = 0.1 and n.minobsinnode = 10.

### 11. Did any of the accuracy measures improve? If so, which ones?

model\_list <- list(original = model\_tree,  
 SMOTE = smote\_fit)  
  
#First Build custom AUC function to extract AUC from the caret model object  
test\_roc <- function(model, data) {  
 roc(data$diagnosis,  
 predict(model, data, type = "prob")[, "M"])  
}  
  
# Get the AUC for the original model with the original dataset.  
model\_tree %>%  
 test\_roc(data = test.data)%>%auc()

## Setting levels: control = B, case = M

## Setting direction: controls < cases

## Area under the curve: 0.9316

# Get the AUC for the SMOTE model with the Imbalanced dataset.  
smote\_fit %>%  
 test\_roc(data = imbal\_test)%>%auc()

## Setting levels: control = B, case = M  
## Setting direction: controls < cases

## Area under the curve: 1

# DATA WRANGLING

### 1. Install and load the library nycflights13

library(tidyverse)  
library(nycflights13)

### 2. Load the datasets/tables flights and airlines

airlines\_data <- airlines  
airports\_data <- airports  
flights\_data <- flights  
planes\_data <- planes  
weather\_data <- weather

### 3. Add full airline name from the airlines table to the flights table that keeps all the records in the flights table by using the appropriate join

### 4. Now add the destination latitude and longitude to the flights table from the airports table by using the appropriate join

# RESAMPLING

Load packages

library(rsample)   
library(purrr)  
library(dplyr)  
library(ggplot2)  
library(scales)

##   
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':  
##   
## discard

## The following object is masked from 'package:readr':  
##   
## col\_factor

library(mlbench)  
library(kernlab)

##   
## Attaching package: 'kernlab'

## The following object is masked from 'package:scales':  
##   
## alpha

## The following object is masked from 'package:purrr':  
##   
## cross

## The following object is masked from 'package:ggplot2':  
##   
## alpha

library(sessioninfo)  
theme\_set(theme\_bw())  
library(RCurl)

Get the German Credit dataset from the UCI machine learning repository

## checking\_account month Credit\_history Purpose Credit\_amount Savings  
## 1 A11 6 A34 A43 1169 A65  
## 2 A12 48 A32 A43 5951 A61  
## 3 A14 12 A34 A46 2096 A61  
## 4 A11 42 A32 A42 7882 A61  
## 5 A11 24 A33 A40 4870 A61  
## 6 A14 36 A32 A46 9055 A65  
## employment Installment\_rate status Other\_debtors Present\_residence\_since  
## 1 A75 4 A93 A101 4  
## 2 A73 2 A92 A101 2  
## 3 A74 2 A93 A101 3  
## 4 A74 2 A93 A103 4  
## 5 A73 3 A93 A101 4  
## 6 A73 2 A93 A101 4  
## Property Age installment\_plans Housing X.credits Job X.people Telephone  
## 1 A121 67 A143 A152 2 A173 1 A192  
## 2 A121 22 A143 A152 1 A173 1 A191  
## 3 A121 49 A143 A152 1 A172 2 A191  
## 4 A122 45 A143 A153 1 A173 2 A191  
## 5 A124 53 A143 A153 2 A173 2 A191  
## 6 A124 35 A143 A153 1 A172 2 A192  
## foreign Cost\_Matrix  
## 1 A201 1  
## 2 A201 2  
## 3 A201 1  
## 4 A201 1  
## 5 A201 2  
## 6 A201 1

## Fit a SVM model to predict the type of credit (good or bad) with the following resampling techniques

#### 5 fold cross validation

* Data preparation, splitting and modelling

# First split the dataset  
set.seed(123)  
training.samples <-credit\_data$Cost\_Matrix %>%   
 createDataPartition(p = 0.8, list = FALSE)  
train.credit<-credit\_data[training.samples, ]  
test.credit<-credit\_data[-training.samples, ]  
  
  
# convert class variable into factor   
credit\_data$Cost\_Matrix=as.factor(credit\_data$Cost\_Matrix)  
  
  
# First split the dataset  
set.seed(123)  
training.samples <-credit\_data$Cost\_Matrix %>%   
 createDataPartition(p = 0.8, list = FALSE)  
train.credit<-credit\_data[training.samples, ]  
test.credit<-credit\_data[-training.samples, ]  
  
#Define training control and modelling  
train\_control <- trainControl(method="cv", number=5)  
model <- train(Cost\_Matrix~., data=train.credit, trControl=train\_control, method="svmLinear")  
# summarize results  
print(model)

## Support Vector Machines with Linear Kernel   
##   
## 800 samples  
## 20 predictor  
## 2 classes: '1', '2'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 640, 640, 640, 640, 640   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.72875 0.3246594  
##   
## Tuning parameter 'C' was held constant at a value of 1

# prediction  
pred=predict(model,newdata = test.credit)  
pred

## [1] 1 2 1 2 2 1 1 2 1 2 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 2 2 2  
## [38] 2 1 1 1 1 2 1 1 1 2 2 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 2 2 2 1 1 1 1 1 1 1  
## [75] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 2 2 1 1 1 2 2  
## [112] 2 1 2 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 2 2 1 1 1 1 1 1 1 2 1 1  
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 1 1 1 1 2 1 2 1 2 1 1 1 1 1 1 2 1 1 1 1 1  
## [186] 2 1 1 2 1 1 2 1 1 2 1 1 1 2 1  
## Levels: 1 2

* Show the accuracy for the training dataset

model$results

## C Accuracy Kappa AccuracySD KappaSD  
## 1 1 0.72875 0.3246594 0.02523329 0.05283238

* Show the accuracy results for each resample

model$resample

## Accuracy Kappa Resample  
## 1 0.70625 0.2964072 Fold1  
## 2 0.73125 0.2973856 Fold2  
## 3 0.75625 0.3709677 Fold3  
## 4 0.70000 0.2682927 Fold4  
## 5 0.75000 0.3902439 Fold5

* Show the accuracy for the test dataset

mean(pred == test.credit$Cost\_Matrix)

## [1] 0.77

* Which resampling method gives the best estimate of the test error?

#### 10 fold cross-validation with 3 repeats

* Modeling and prediction

# define training control  
train\_control2 <- trainControl(method="repeatedcv", number=10, repeats=3)  
model2<-train(Cost\_Matrix~.,data=train.credit,trControl=train\_control2, method="svmLinear")  
# summarize results  
print(model2)

## Support Vector Machines with Linear Kernel   
##   
## 800 samples  
## 20 predictor  
## 2 classes: '1', '2'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 720, 720, 720, 720, 720, 720, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.7420833 0.3454704  
##   
## Tuning parameter 'C' was held constant at a value of 1

pred2=predict(model2,newdata = test.credit)  
pred2

## [1] 1 2 1 2 2 1 1 2 1 2 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 2 2 2  
## [38] 2 1 1 1 1 2 1 1 1 2 2 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 2 2 2 1 1 1 1 1 1 1  
## [75] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 2 2 1 1 1 2 2  
## [112] 2 1 2 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 2 2 1 1 1 1 1 1 1 2 1 1  
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 1 1 1 1 2 1 2 1 2 1 1 1 1 1 1 2 1 1 1 1 1  
## [186] 2 1 1 2 1 1 2 1 1 2 1 1 1 2 1  
## Levels: 1 2

* Show the accuracy for the training dataset

model2$results

## C Accuracy Kappa AccuracySD KappaSD  
## 1 1 0.7420833 0.3454704 0.03973014 0.1014029

* Show the accuracy results for each resample

model2$resample

## Accuracy Kappa Resample  
## 1 0.7250 0.2567568 Fold01.Rep1  
## 2 0.7125 0.3072289 Fold02.Rep1  
## 3 0.7875 0.5000000 Fold03.Rep1  
## 4 0.7500 0.3902439 Fold04.Rep1  
## 5 0.7125 0.2901235 Fold05.Rep1  
## 6 0.7125 0.2333333 Fold06.Rep1  
## 7 0.7500 0.3243243 Fold07.Rep1  
## 8 0.8000 0.5238095 Fold08.Rep1  
## 9 0.7750 0.4078947 Fold09.Rep1  
## 10 0.7000 0.1891892 Fold10.Rep1  
## 11 0.7500 0.4047619 Fold01.Rep2  
## 12 0.6750 0.2441860 Fold02.Rep2  
## 13 0.7375 0.2808219 Fold03.Rep2  
## 14 0.7375 0.3000000 Fold04.Rep2  
## 15 0.7875 0.4480519 Fold05.Rep2  
## 16 0.7375 0.3354430 Fold06.Rep2  
## 17 0.6750 0.1666667 Fold07.Rep2  
## 18 0.8125 0.5370370 Fold08.Rep2  
## 19 0.7625 0.3987342 Fold09.Rep2  
## 20 0.7375 0.3181818 Fold10.Rep2  
## 21 0.7375 0.3181818 Fold01.Rep3  
## 22 0.7125 0.2901235 Fold02.Rep3  
## 23 0.8250 0.5625000 Fold03.Rep3  
## 24 0.7125 0.2901235 Fold04.Rep3  
## 25 0.7750 0.3571429 Fold05.Rep3  
## 26 0.7500 0.3589744 Fold06.Rep3  
## 27 0.7500 0.3421053 Fold07.Rep3  
## 28 0.7875 0.4620253 Fold08.Rep3  
## 29 0.6875 0.2283951 Fold09.Rep3  
## 30 0.6875 0.2977528 Fold10.Rep3

* Show the accuracy for the test dataset

mean(pred2 == test.credit$Cost\_Matrix)

## [1] 0.77

* Which resampling method gives the best estimate of the test error?

#### Leave group out with 5 iterations and 85% data used for training define training control

* Split the dataset 85% training and 15% testing, modeling

set.seed(123)  
training.samples <-credit\_data$Cost\_Matrix %>%   
 createDataPartition(p = 0.85, list = FALSE)  
train.data<-credit\_data[training.samples, ]  
test.data<-credit\_data[-training.samples, ]  
  
# Modelling  
train\_control3 <- trainControl(method="LOOCV")  
# train the model  
model3<-train(Cost\_Matrix~.,data=train.data,trControl=train\_control3, method="svmLinear")  
# summarize results  
print(model3)

## Support Vector Machines with Linear Kernel   
##   
## 850 samples  
## 20 predictor  
## 2 classes: '1', '2'   
##   
## No pre-processing  
## Resampling: Leave-One-Out Cross-Validation   
## Summary of sample sizes: 849, 849, 849, 849, 849, 849, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.7447059 0.3522388  
##   
## Tuning parameter 'C' was held constant at a value of 1

* Prediction

pred3=predict(model3,newdata=test.data)  
pred3

## [1] 1 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 2 1 1 1 1 2 1 1 1 1  
## [38] 1 1 1 2 1 2 1 1 1 2 2 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1  
## [75] 1 1 1 1 1 2 1 1 1 1 1 2 1 1 2 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 2 1 2 1 1 2  
## [112] 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 2 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 2 2 2 1  
## [149] 1 1  
## Levels: 1 2

* Show the accuracy for the training dataset

model3$results

## C Accuracy Kappa  
## 1 1 0.7447059 0.3522388

* Show the accuracy results for each resample

model3$resample

## NULL

* Show the accuracy for the test dataset

mean(pred3 == test.credit$Cost\_Matrix)

## [1] 0.66

* Which resampling method gives the best estimate of the test error?

#### Bootstrap with 25 iterations

* Modeling and prediction

# Bootstrap with 25 iterations  
# define training control  
train\_control4 <- trainControl(method="boot", number=25)  
# train the model  
model4 <- train(Cost\_Matrix~.,data=train.credit,trControl=train\_control4,method="svmLinear")  
# summarize results  
print(model4)

## Support Vector Machines with Linear Kernel   
##   
## 800 samples  
## 20 predictor  
## 2 classes: '1', '2'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 800, 800, 800, 800, 800, 800, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.7234505 0.3090998  
##   
## Tuning parameter 'C' was held constant at a value of 1

# Prediction  
pred4=predict(model4,newdata=test.credit)  
pred4

## [1] 1 2 1 2 2 1 1 2 1 2 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 2 2 2  
## [38] 2 1 1 1 1 2 1 1 1 2 2 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 2 2 2 1 1 1 1 1 1 1  
## [75] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 2 2 1 1 1 2 2  
## [112] 2 1 2 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 2 2 1 1 1 1 1 1 1 2 1 1  
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 1 1 1 1 2 1 2 1 2 1 1 1 1 1 1 2 1 1 1 1 1  
## [186] 2 1 1 2 1 1 2 1 1 2 1 1 1 2 1  
## Levels: 1 2

* Show the accuracy for the training dataset

model4$results

## C Accuracy Kappa AccuracySD KappaSD  
## 1 1 0.7234505 0.3090998 0.02226505 0.05945441

* Show the accuracy results for each resample

model4$resample

## Accuracy Kappa Resample  
## 1 0.7299035 0.3083766 Resample01  
## 2 0.7296417 0.2990675 Resample02  
## 3 0.7368421 0.3891985 Resample03  
## 4 0.6833333 0.2196057 Resample04  
## 5 0.7108844 0.2802005 Resample05  
## 6 0.7142857 0.3128834 Resample06  
## 7 0.7474048 0.3303603 Resample07  
## 8 0.6966667 0.2448551 Resample08  
## 9 0.7591973 0.3581778 Resample09  
## 10 0.7374101 0.3473339 Resample10  
## 11 0.6962457 0.2539410 Resample11  
## 12 0.6777409 0.2009360 Resample12  
## 13 0.7185430 0.3426376 Resample13  
## 14 0.7552448 0.3844546 Resample14  
## 15 0.7442623 0.3815317 Resample15  
## 16 0.7216495 0.2462345 Resample16  
## 17 0.7214765 0.3513244 Resample17  
## 18 0.7375415 0.3716407 Resample18  
## 19 0.6794872 0.1580311 Resample19  
## 20 0.7242525 0.3272022 Resample20  
## 21 0.7333333 0.3228373 Resample21  
## 22 0.7404844 0.3100430 Resample22  
## 23 0.7241379 0.3349005 Resample23  
## 24 0.7353952 0.3241336 Resample24  
## 25 0.7308970 0.3275876 Resample25

* Show the accuracy for the test dataset

mean(pred4 == test.credit$Cost\_Matrix)

## [1] 0.77

# Which resampling method gives the best estimate of the test error?