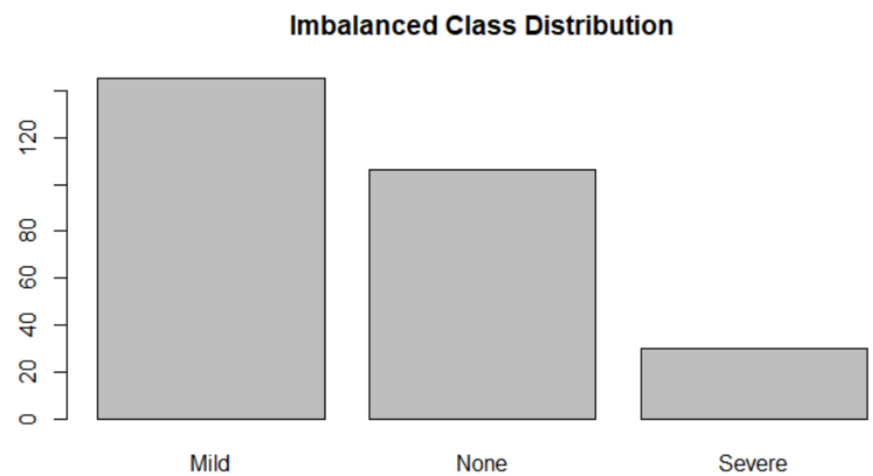
**Homework – 5**

**Question 12.1**

(a) Given the classification imbalance in hepatic injury status, describe how you would create a training and testing set.
**Answer:** On observing the hepatic dataset, we observe that there is a huge imbalance in injury status. In order to overcome this, we can use stratified random sampling approach. Below graph shows the imbalance between the three injury classes.




(b) Which classification statistic would you choose to optimize for this exercise and why?
**Answer**: For this exercise, we will be using "Accuracy" as the classification statistic as Accuracy is the percentage of correctly classifies instances out of all instances.

(c) Split the data into a training and a testing set, pre-process the data, and build models described in this chapter for the biological predictors. Using each model to predict on the testing set, which model has the best predictive ability for the biological predictors and what is the optimal performance?

**Answer:** We split our data into 75% training and 25% as testing data. Before splitting the data, we preprocess it by removing near-zero variance predictors, center and scale, and by removing highly correlated predictors.

**Pre-Process**:

```
#PreProcess the data

#Removing nearzero variance
NV <- nearZeroVar(bio)
NV
noZVbio <- bio[,-NV]
noZVbio

#Calculating missing values
sum(is.na(noZVbio))

#Finding correlation
set.seed(1)
highCorBio<-findCorrelation(cor(noZVbio),cutoff = .80)
filteredCorBio <- noZVbio[,-highCorBio]
```

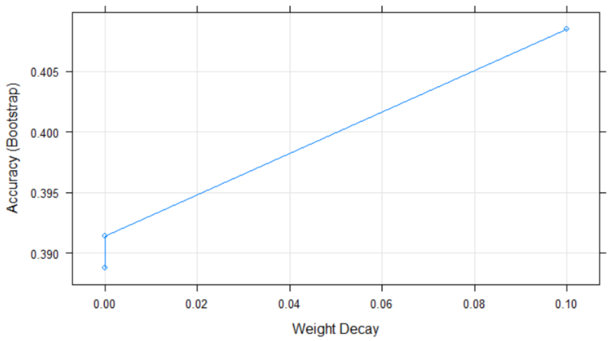
### Splitting Data:

```
# Split the data

set.seed(1)

trainingRows =  createDataPartition(injury, p = .75, list= FALSE)

trainBio <- filteredCorBio[ trainingRows, ]
testBio <- filteredCorBio[-trainingRows, ]

trainInjury <- injury[trainingRows]
testInjury <- injury[-trainingRows]
```

## Logistic Regression Model:

```
Confusion Matrix and Statistics

          Reference
Prediction Mild None Severe
    Mild     21    9      3
    None     14   11      1
    Severe    1    6      3

Overall Statistics

               Accuracy : 0.5072
                 95% CI : (0.3841, 0.6298)
    No Information Rate : 0.5217
    P-Value [Acc > NIR] : 0.6416

                  Kappa : 0.1701

 Mcnemar's Test P-Value : 0.1295

Statistics by Class:

                     Class: Mild Class: None Class: Severe
Sensitivity               0.5833      0.4231       0.42857
Specificity               0.6364      0.6512       0.88710
Pos Pred Value            0.6364      0.4231       0.30000
Neg Pred Value            0.5833      0.6512       0.93220
Prevalence                0.5217      0.3768       0.10145
Detection Rate            0.3043      0.1594       0.04348
Detection Prevalence      0.4783      0.3768       0.14493
Balanced Accuracy         0.6098      0.5371       0.65783
```

## Plot for Logistic Regression:




## Linear Discriminant Analysis Model:

```
Confusion Matrix and Statistics

          Reference
Prediction Mild None Severe
    Mild     20    9      6
    None     14   14      0
    Severe    2    3      1

Overall Statistics

               Accuracy : 0.5072
                 95% CI : (0.3841, 0.6298)
    No Information Rate : 0.5217
    P-Value [Acc > NIR] : 0.6416

                  Kappa : 0.141

 Mcnemar's Test P-Value : 0.1075

Statistics by Class:

                     Class: Mild Class: None Class: Severe
Sensitivity               0.5556      0.5385       0.14286
Specificity               0.5455      0.6744       0.91935
Pos Pred Value            0.5714      0.5000       0.16667
Neg Pred Value            0.5294      0.7073       0.90476
Prevalence                0.5217      0.3768       0.10145
Detection Rate            0.2899      0.2029       0.01449
Detection Prevalence      0.5072      0.4058       0.08696
Balanced Accuracy         0.5505      0.6064       0.53111
```

**Partial Least Square Discriminant Analysis Model:**

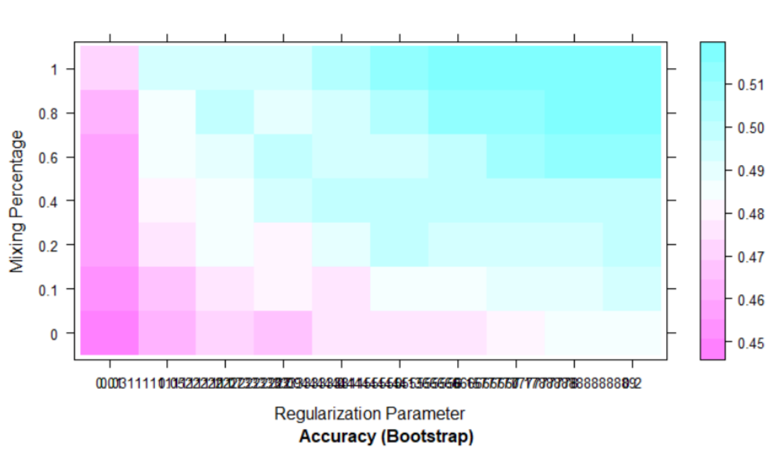
```
Confusion Matrix and Statistics

          Reference
Prediction Mild None Severe
    Mild     27   16      7
    None      9   10      0
    Severe    0    0      0

Overall Statistics

               Accuracy : 0.5362
                 95% CI : (0.412, 0.6572)
    No Information Rate : 0.5217
    P-Value [Acc > NIR] : 0.4528

                  Kappa : 0.105

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Mild Class: None Class: Severe
Sensitivity               0.7500      0.3846        0.0000
Specificity               0.3030      0.7907        1.0000
Pos Pred Value            0.5400      0.5263           NaN
Neg Pred Value            0.5263      0.6800        0.8986
Prevalence                0.5217      0.3768        0.1014
Detection Rate            0.3913      0.1449        0.0000
Detection Prevalence      0.7246      0.2754        0.0000
Balanced Accuracy         0.5265      0.5877        0.5000
```

**Penalized Model:**

```
Confusion Matrix and Statistics

          Reference
Prediction Mild None Severe
    Mild     30   17      7
    None      6    9      0
    Severe    0    0      0

Overall Statistics

               Accuracy : 0.5652
                 95% CI : (0.4404, 0.6842)
    No Information Rate : 0.5217
    P-Value [Acc > NIR] : 0.274

                  Kappa : 0.1471

 Mcnemar's Test P-Value : NA
```

```
Statistics by Class:

                     Class: Mild Class: None Class: Severe
Sensitivity               0.8333      0.3462        0.0000
Specificity               0.2727      0.8605        1.0000
Pos Pred Value            0.5556      0.6000           NaN
Neg Pred Value            0.6000      0.6852        0.8986
Prevalence                0.5217      0.3768        0.1014
Detection Rate            0.4348      0.1304        0.0000
Detection Prevalence      0.7826      0.2174        0.0000
Balanced Accuracy         0.5530      0.6033        0.5000
```

**Plot of Penalized Model:**

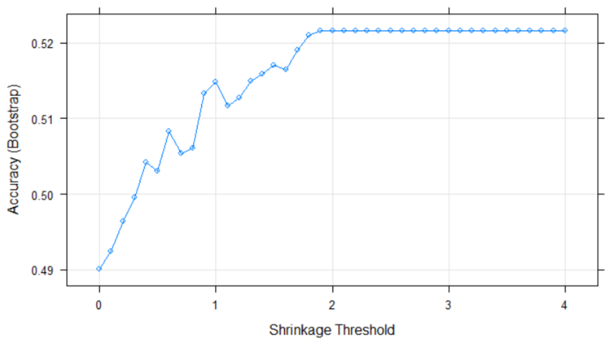



## Nearest Shrunken Centroids:

```
Confusion Matrix and Statistics

          Reference
Prediction Mild None Severe
    Mild     36   26      7
    None      0    0      0
    Severe    0    0      0

Overall Statistics

               Accuracy : 0.5217
                 95% CI : (0.398, 0.6435)
    No Information Rate : 0.5217
    P-Value [Acc > NIR] : 0.5486

                  Kappa : 0

 Mcnemar's Test P-Value : NA
```

```
Statistics by Class:

                     Class: Mild Class: None Class: Severe
Sensitivity               1.0000      0.0000        0.0000
Specificity               0.0000      1.0000        1.0000
Pos Pred Value            0.5217         NaN           NaN
Neg Pred Value               NaN      0.6232        0.8986
Prevalence                0.5217      0.3768        0.1014
Detection Rate            0.5217      0.0000        0.0000
Detection Prevalence      1.0000      0.0000        0.0000
Balanced Accuracy         0.5000      0.5000        0.5000
```

### Plot of Nearest Shrunken Centroids:

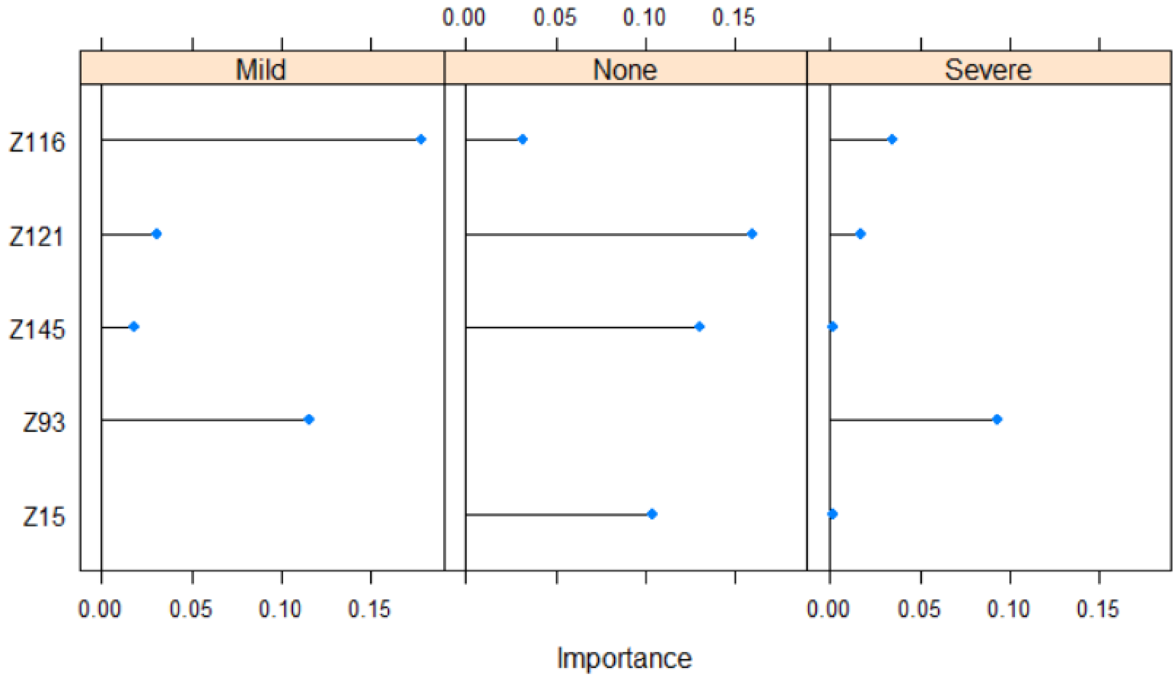



| Model | Accuracy |
|---|---|
| Logistic Regression | 0.5072 |
| Linear Discriminant Analysis | 0.5072 |
| Partial Least Square Discriminant Analysis | 0.5362 |
| Penalized Model | 0.5652 |
| Nearest Shrunken Centroids | 0.5217 |

Out of all the models **Penalized Model** has the maximum accuracy of **0.5652**.

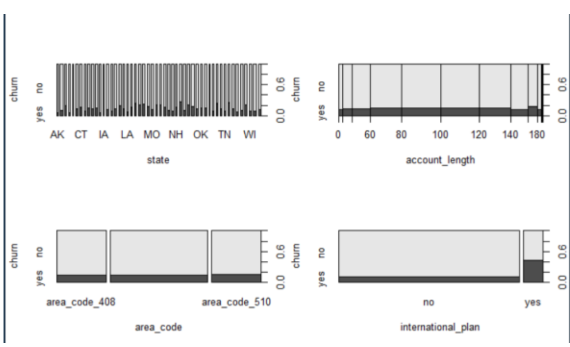
(d) For the optimal model for the biological predictors, what are the top five important predictors?

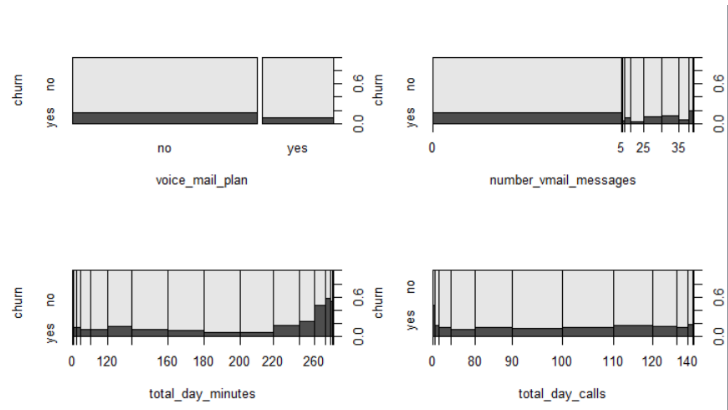
**Answer:** The top five important predictors of Penalized Model are:

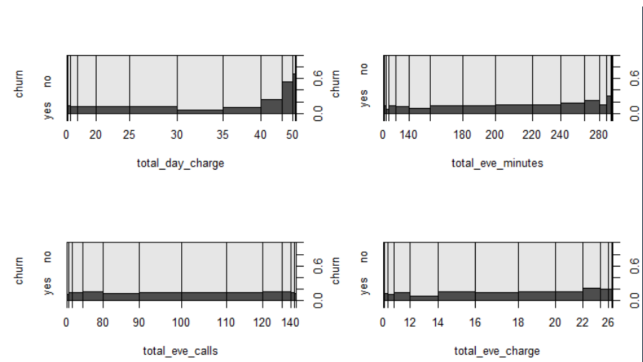


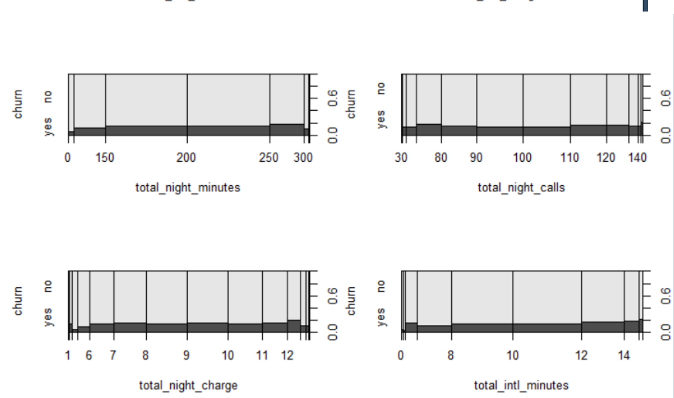

**Question 12.3**

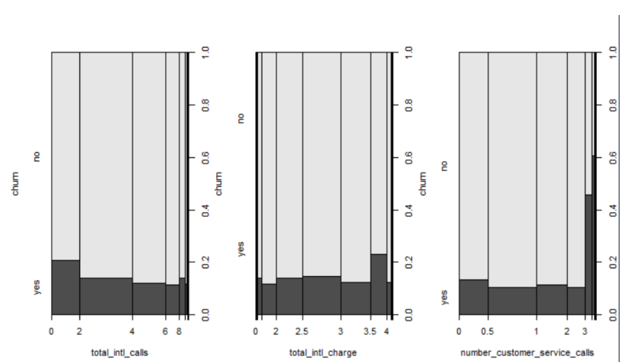
(a) Explore the data by visualizing the relationship between the predictors and the outcome. Are there important features of the predictor data themselves, such as between-predictor correlations or degenerate distributions?
Answer: We plot each predictor against the outcome to understand the relationship between them:



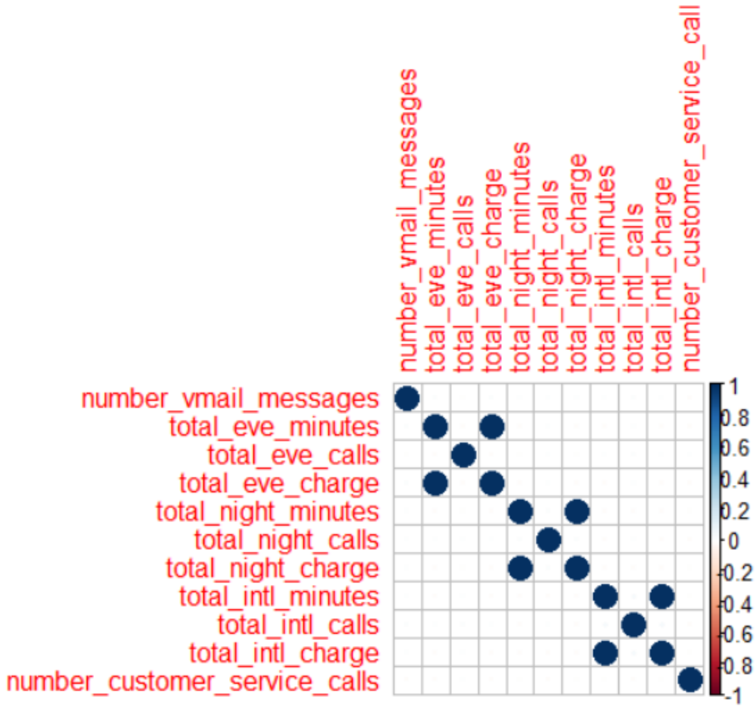



We also find the correlation between the predictors using the corrplot.




(b) What criteria should be used to evaluate the effectiveness of the models?

**Answer:** For this question, we will be using "**ROC**" as the classification statistic as we are using **LGOCV** as the resampling technique.

(c) Fit models covered in class to the training set and tune them via resampling. Which model has the best performance?

**Answer:**

**Pre-Process**: We remove near zero variance predictors, center and scale the data, correct the skewness using BoxCox and remove highly correlated data.

```
#PreProcess

newdata <- preProcess(predict_train, method = c("center","scale","BoxCox"))
newdata_train <- predict(newdata, predict_train)
newdata_test <- predict(newdata, predict_test)


NV3 <- nearZeroVar(newdata_train)
newdata_train <- newdata_train[-NV3]
newdata_test <- newdata_test[-NV3]

highcor <- cor(newdata_train)
highcorpredict <- findCorrelation(highcor)

newdata_train <- newdata_train[,-highcorpredict]
newdata_test <- newdata_test[,-highcorpredict]
```

**Logistic Regression Model:**

```
Generalized Linear Model

3333 samples
  14 predictor
   2 classes: 'yes', 'no'

Pre-processing: centered (14), scaled (14)
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 2501, 2501, 2501, 2501, 2501, 2501, ...
Resampling results:

  ROC        Sens  Spec
  0.5178488  0     1
```

**Linear Discriminant Analysis Model:**

```
> lda
Linear Discriminant Analysis

3333 samples
  14 predictor
   2 classes: 'yes', 'no'

Pre-processing: centered (14), scaled (14)
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 2501, 2501, 2501, 2501, 2501, 2501, ...
Resampling results:

  ROC        Sens  Spec
  0.5179583  0     1
```

**Partial Least Square Discriminant Analysis Model:**

```
> plsFitc
Partial Least Squares

3333 samples
  14 predictor
   2 classes: 'yes', 'no'

Pre-processing: centered (14), scaled (14)
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 2501, 2501, 2501, 2501, 2501, 2501, ...
Resampling results:

  ROC        Sens  Spec
  0.5200023  0     1

Tuning parameter 'ncomp' was held constant at a value of 1
```

**Penalized Model:**

```
Resampling results across tuning parameters:

  alpha  lambda      ROC        Sens  Spec
  0.0    0.01000000  0.5181737  0     1
  0.0    0.03111111  0.5185520  0     1
  0.0    0.05222222  0.5188450  0     1
  0.0    0.07333333  0.5190398  0     1
  0.0    0.09444444  0.5191742  0     1
  0.0    0.11555556  0.5192664  0     1
  0.0    0.13666667  0.5193272  0     1
  0.0    0.15777778  0.5194139  0     1
  0.0    0.17888889  0.5194448  0     1
  0.0    0.20000000  0.5195239  0     1
  0.1    0.01000000  0.5178146  0     1
  0.1    0.03111111  0.5144537  0     1
  0.1    0.05222222  0.5083886  0     1
  0.1    0.07333333  0.5025625  0     1
  0.1    0.09444444  0.4981894  0     1
  0.1    0.11555556  0.4921711  0     1
  0.1    0.13666667  0.4934438  0     1
  0.1    0.15777778  0.4943153  0     1
  0.1    0.17888889  0.4984586  0     1
  0.1    0.20000000  0.5000000  0     1
  0.2    0.01000000  0.5164733  0     1
  0.2    0.03111111  0.5056959  0     1
  0.2    0.05222222  0.4960028  0     1
  0.2    0.07333333  0.4930243  0     1
  0.2    0.09444444  0.4984841  0     1
  0.2    0.11555556  0.5000000  0     1
  0.2    0.13666667  0.5000000  0     1
  0.2    0.15777778  0.5000000  0     1
  0.2    0.17888889  0.5000000  0     1
  0.2    0.20000000  0.5000000  0     1
  0.4    0.01000000  0.5119995  0     1
  0.4    0.03111111  0.4926297  0     1
  0.4    0.05222222  0.5000000  0     1
  0.4    0.07333333  0.5000000  0     1
  0.4    0.09444444  0.5000000  0     1
  0.4    0.11555556  0.5000000  0     1
  0.4    0.13666667  0.5000000  0     1
  0.4    0.15777778  0.5000000  0     1
  0.4    0.17888889  0.5000000  0     1
  0.4    0.20000000  0.5000000  0     1
  0.6    0.01000000  0.5061402  0     1
  0.6    0.03111111  0.4984841  0     1
  0.6    0.05222222  0.5000000  0     1
  0.6    0.07333333  0.5000000  0     1
  0.6    0.09444444  0.5000000  0     1
  0.6    0.11555556  0.5000000  0     1
  0.6    0.13666667  0.5000000  0     1
  0.6    0.15777778  0.5000000  0     1
  0.6    0.17888889  0.5000000  0     1
  0.6    0.20000000  0.5000000  0     1
  0.8    0.01000000  0.5010257  0     1
  0.8    0.03111111  0.5000000  0     1
  0.8    0.05222222  0.5000000  0     1
  0.8    0.07333333  0.5000000  0     1
  0.8    0.09444444  0.5000000  0     1
  0.8    0.11555556  0.5000000  0     1
  0.8    0.13666667  0.5000000  0     1
  0.8    0.15777778  0.5000000  0     1
```

```
  0.8    0.17888889  0.5000000  0      1
  0.8    0.20000000  0.5000000  0      1
  1.0    0.01000000  0.4961723  0      1
  1.0    0.03111111  0.5000000  0      1
  1.0    0.05222222  0.5000000  0      1
  1.0    0.07333333  0.5000000  0      1
  1.0    0.09444444  0.5000000  0      1
  1.0    0.11555556  0.5000000  0      1
  1.0    0.13666667  0.5000000  0      1
  1.0    0.15777778  0.5000000  0      1
  1.0    0.17888889  0.5000000  0      1
  1.0    0.20000000  0.5000000  0      1

ROC was used to select the optimal model using the largest value.
The final values used for the model were alpha = 0 and lambda = 0.2.
```

**Nearest Shrunken Centroids:**

```
> nscTunedc
Nearest Shrunken Centroids

3333 samples
  14 predictor
   2 classes: 'yes', 'no'

Pre-processing: centered (14), scaled (14)
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 2501, 2501, 2501, 2501, 2501, 2501, ...
Resampling results across tuning parameters:

  threshold  ROC        Sens  Spec
  0.0        0.5199949  0     1
  0.1        0.5181676  0     1
  0.2        0.5150843  0     1
  0.3        0.5111367  0     1
  0.4        0.5068965  0     1
  0.5        0.5031414  0     1
  0.6        0.5004422  0     1
  0.7        0.4961718  0     1
  0.8        0.4921142  0     1
  0.9        0.4939239  0     1
  1.0        0.4933488  0     1
  1.1        0.4942849  0     1
  1.2        0.4967039  0     1
  1.3        0.4984841  0     1
  1.4        0.5000000  0     1
  1.5        0.5000000  0     1
  1.6        0.5000000  0     1
  1.7        0.5000000  0     1
  1.8        0.5000000  0     1
  1.9        0.5000000  0     1
  2.0        0.5000000  0     1
  2.1        0.5000000  0     1
  2.2        0.5000000  0     1
  2.3        0.5000000  0     1
  2.4        0.5000000  0     1
  2.5        0.5000000  0     1
```

```
  2.6         0.5000000  0     1
  2.7         0.5000000  0     1
  2.8         0.5000000  0     1
  2.9         0.5000000  0     1
  3.0         0.5000000  0     1
  3.1         0.5000000  0     1
  3.2         0.5000000  0     1
  3.3         0.5000000  0     1
  3.4         0.5000000  0     1
  3.5         0.5000000  0     1
  3.6         0.5000000  0     1
  3.7         0.5000000  0     1
  3.8         0.5000000  0     1
  3.9         0.5000000  0     1
  4.0         0.5000000  0     1

ROC was used to select the optimal model using the largest value.
The final value used for the model was threshold = 0.
```

| Model | ROC |
| --- | --- |
| Logistic Regression | 0.5178 |
| Linear Discriminant Analysis | 0.5179 |
| Partial Least Square Discriminant Analysis | 0.5200 |
| Penalized Model | 0.5195 |
| Nearest Shrunken Centroids | 0.5199 |

The best model out of all of them is **Partial Least Square Discriminant Analysis (PLSDA).**

**RCODE:**

**install.packages(c("glmnet", "pamr", "rms", "sparseLDA", "subselect"))**

**#12.1**

**library(caret)**

**library(AppliedPredictiveModeling)**

```
data(hepatic)

library(MASS)
set.seed(1)

barplot(table(injury), main="Imbalanced Class Distribution")

#PreProcess the data

#Removing nearzero variance
NV <- nearZeroVar(bio)
NV
noZVbio <- bio[,-NV]
noZVbio

#Calculating missing values
sum(is.na(noZVbio))

#Finding correlation
set.seed(1)
highCorBio<-findCorrelation(cor(noZVbio),cutoff = .80)
filteredCorBio <- noZVbio[,-highCorBio]
```

```
# Split the data

set.seed(1)

trainingRows =  createDataPartition(injury, p = .75, list= FALSE)

trainBio <- filteredCorBio[ trainingRows, ]
testBio <- filteredCorBio[-trainingRows, ]

trainInjury <- injury[trainingRows]
testInjury <- injury[-trainingRows]

#Model building

##Multinomial Logistic Regression##

set.seed(1)
ctrl <- trainControl(summaryFunction = defaultSummary)
lrBio <- train(x=trainBio,
            y = trainInjury,
            method = "multinom",
            preProc = c("center", "scale"),
            metric = "Accuracy",
```

```
             trControl = ctrl)
summary(lrBio)

plot(lrBio)


predictionLRBio<-predict(lrBio,testBio)

confusionMatrix(data =predictionLRBio,
              reference = testInjury)


##Linear Discriminant Analysis

set.seed(1)

ldaBio <- train(x = trainBio,
              y = trainInjury,
              method = "lda",
              preProc = c("center", "scale"),
              metric = "Accuracy",
              trControl = ctrl)
```

```
summary(ldaBio)



predictionLDABio <- predict(ldaBio,testBio)
confusionMatrix(data =predictionLDABio,
          reference = testInjury)



############## Partial Least Squares Discriminant Analysis ###############
library(MASS)
set.seed(1)
plsFit <- train(x = trainBio,
          y = trainInjury,
          method = "pls",
          tuneGrid = expand.grid(.ncomp = 1:1),
          preProc = c("center","scale"),
          metric = "Accuracy",
          trControl = ctrl)

plsFit

plot(plsFit)
summary(plsFit)
```

```
varImp(plsFit, scale = FALSE)


predictionPLSBio <-predict(plsFit,testBio)

confusionMatrix(data =predictionPLSBio,

                reference = testInjury)


############ Penalized Models ##########

## The primary package for penalized logistic regression is glmnet.


library(caret)

set.seed(1)

ctrl1 <- trainControl(method = "cv", number = 10)

glmnGrid <- expand.grid(.alpha = c(0, .1, .2, .4, .6, .8, 1),

                        .lambda = seq(.01, .2, length = 10))


glmnTuned <- train(x = trainBio, y = trainInjury, method = "glmnet",

                   tuneGrid = glmnGrid,

                   preProc = c("center", "scale"),

                   metric = "Accuracy", trControl = ctrl1)


summary(glmnTuned)


important <- varImp(glmnTuned, scale = FALSE)

plot(important, top = 5, scales = list(y = list(cex = .95)))
```

```
predictGlmnetBio <-  predict(glmnTuned,testBio)
confusionMatrix(data = predictGlmnetBio,
                reference = testInjury)

plot(glmnTuned, plotType = "level")




########### Nearest Shrunken Centroids ###########


library(pamr)


nscGrid <- data.frame(.threshold = seq(0,4, by=0.1))
set.seed(1)
nscTuned <- train(x = trainBio, y = trainInjury, method = "pam",
                  preProc = c("center", "scale"), tuneGrid = nscGrid,
                  metric = "Accuracy", trControl = ctrl)
nscTuned


plot(nscTuned)


summary(nscTuned)
```

```
predictNSC <-predict(nscTuned,testBio)

confusionMatrix(data =predictNSC,

                reference = testInjury)


predictors(nscTuned)





#12.3


#12.3


library(C50)

library(corrplot)

data(churn)


str(churnTrain)


table(churnTrain$churn)


#plot


par(mfrow = c(2,2))
```

```
plot(churn~state,data = churnTrain)
plot(churn~account_length,data = churnTrain)
plot(churn~area_code,data = churnTrain)
plot(churn~international_plan,data = churnTrain)
par(mfrow = c(2,2))
plot(churn~voice_mail_plan,data = churnTrain)
plot(churn~number_vmail_messages,data = churnTrain)
plot(churn~total_day_minutes,data = churnTrain)
plot(churn~total_day_calls,data = churnTrain)
par(mfrow = c(2,2))
plot(churn~total_day_charge,data = churnTrain)
plot(churn~total_eve_minutes,data = churnTrain)
plot(churn~total_eve_calls,data = churnTrain)
plot(churn~total_eve_charge,data = churnTrain)
par(mfrow = c(2,2))
plot(churn~total_night_minutes,data = churnTrain)
plot(churn~total_night_calls,data = churnTrain)
plot(churn~total_night_charge,data = churnTrain)
plot(churn~total_intl_minutes,data = churnTrain)
par(mfrow = c(1,3))
plot(churn~total_intl_calls,data = churnTrain)
plot(churn~total_intl_charge,data = churnTrain)
plot(churn~number_customer_service_calls,data = churnTrain)
```

```
#(c)

predict_train <- churnTrain[,-20]
ctrain <- churnTrain[,20]
predict_test <- churnTest[,-20]
ctest <- churnTest[,20]

#Dummy Variables

library(caret)

dummy <- dummyVars("~state + area_code + international_plan + voice_mail_plan",
             data = predict_train, fullRank = TRUE)
dummytrain <- data.frame(predict(dummy, newdata = predict_train))
dummy <- dummyVars("~state + area_code + international_plan + voice_mail_plan",
             data = predict_test, fullRank = TRUE)
dummytest <- data.frame(predict(dummy, newdata = predict_test))


# Drop all factor predictors:

predict_train <- predict_train[,-c(1,3,4,5)]
```

```r
predict_test <- predict_test[,-c(1,3,4,5)]

predict_train <- merge(predict_train, dummytrain, by =0)
predict_test <- merge(predict_test, dummytest, by =0)

predict_train <- predict_train[,-c(1)]
predict_test <- predict_test[,-c(1)]

#PreProcess

newdata <- preProcess(predict_train, method = c("center","scale","BoxCox"))
newdata_train <- predict(newdata, predict_train)
newdata_test <- predict(newdata, predict_test)


NV3 <- nearZeroVar(newdata_train)
newdata_train <- newdata_train[-NV3]
newdata_test <- newdata_test[-NV3]

highcor <- cor(newdata_train)
highcorpredict <- findCorrelation(highcor)

newdata_train <- newdata_train[,-highcorpredict]
newdata_test <- newdata_test[,-highcorpredict]
```

```
library(pROC)

ctrl1 = trainControl(method = "LGOCV",

                    summaryFunction=twoClassSummary,

                    classProbs=TRUE )


# Logistic Regression Model:


set.seed(1)

lrnew <- train(x=newdata_train,

               y = ctrain,

               method = "glm",

               preProc = c("center", "scale"),

               metric = "ROC",

               trControl = ctrl1)

summary(lrnew)


lrnew



predictionLR<-predict(lrnew,newdata_test)
```

```
confusionMatrix(data =predictionLR,
        reference = ctest)



# Linear Discriminant Analysis:

set.seed(1)
lda <- train(x = newdata_train,
        y = ctrain,
        method = "lda",
        preProc = c("center", "scale"),
        metric = "ROC",
        trControl = ctrl1)

lda
summary(lda)


predictionLDA <- predict(lda,newdata_test)
confusionMatrix(data =predictionLDA,
        reference = ctest)
```

```
############## Partial Least Squares Discriminant Analysis ###############

library(MASS)

set.seed(1)

plsFitc <- train(x = newdata_train,

          y = ctrain,

          method = "pls",

          tuneGrid = expand.grid(.ncomp = 1:1),

          preProc = c("center","scale"),

          metric = "ROC",

          trControl = ctrl1)


plsFitc


summary(plsFitc)


varImp(plsFitc, scale = FALSE)


predictionPLS <-predict(plsFitc,newdata_test)

confusionMatrix(data =predictionPLS,

          reference = ctest)


# Penalized Methods:


library(caret)
```

```
set.seed(1)

glmnGrid <- expand.grid(.alpha = c(0, .1, .2, .4, .6, .8, 1),

                        .lambda = seq(.01, .2, length = 10))


glmnTuned <- train(x = newdata_train, y = ctrain, method = "glmnet",

                   tuneGrid = glmnGrid,

                   preProc = c("center", "scale"),

                   metric = "ROC", trControl = ctrl1)


glmnTuned


important <- varImp(glmnTuned, scale = FALSE)

plot(important, top = 5, scales = list(y = list(cex = .95)))



predictGlmnet <-  predict(glmnTuned,newdata_test)

confusionMatrix(data = predictGlmnet,

                reference = ctest)


plot(glmnTuned, plotType = "level")


# Nearest shrunken Centroids:
```

```
library(pamr)

nscGrid <- data.frame(.threshold = seq(0,4, by=0.1))
set.seed(1)
nscTunedc <- train(x = newdata_train, y = ctrain, method = "pam",
                   preProc = c("center", "scale"), tuneGrid = nscGrid,
                   metric = "ROC", trControl = ctrl1)
nscTunedc

plot(nscTunedc)

summary(nscTunedc)

predictNSCc <-predict(nscTunedc,newdata_test)
confusionMatrix(data =predictNSCc,
                 reference = ctest)

predictors(nscTunedc)
```