**R Code Included in Text**

**Chapter 3**

**Appendix 3.10: R Code for Grouping  
2017 Medical and Surgical MS-DRG Codes**

# before running this script, make sure data is a data table, column with MS-DRG value is in string format

# DRG is composed of 3 groups: medical, surgical, and ungrouped (invalid DRG code)

# source of this classification could be found here:

# <https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/MedicareFeeforSvcPartsAB/downloads/DRGdesc08.pdf>

#input values of DRG for surgical:

surgicalG<- c(1:42,113:117,129:139,163:168,215:274,326:358,405:425,

453:520,570:585,614:630,652:675,707:718,734:750,765:770,789:804,820:830,853:858,876,901:909,927:929,939:941,955:959,969:970,981:989)

#input values of DRG for medical:

medicalG<- c(52:103,121:125,146:159,175:208,280:316,368:395,432:446,

533:566,592:607,637:645,682:700,722:730,754:761,774:782,808:816,834:849,862:872,880:897,913:923,933:935,945:951,963:965,974:977)

data<- as.data.table(data)

# make sure DRG.Code is in string format:

data[,DRG.Code:=as.character(DRG.Code)]

# create a class column for DRG code:

data[DRG.Code %in% surgicalG,DRG.Class:=”surg”]

data[DRG.Code %in% edical,DRG.Class:=”med”]

data[is.na(DRG.Class),DRG.Class:=”ungroup”]

**Note:** this code does not classify MS-DRG 998 (invalid code) or 999 (ungroupable); these codes are unclassified as either medical or surgical and must be addressed separately in any analysis.

**Chapter 8**

**Appendix 2: R code for linear regression model**

## Install backing packages

install.packages("ggplot2")

install.packages("data.table")

install.packages("scales")

install.packages("dplyr")

install.packages("MASS")

## Load the libraries we just installed

library(ggplot2)

library(data.table)

library(scales)

library(dplyr)

library(MASS)

##================================================================

## Data Pre-processing

##================================================================

## Reading in the dataset

data = read.csv("/C/Users/User1/Documents/Blue\_book\_sample\_data\_updated.csv", # file location on your computer

header=T)

## Convert data to a data.table object

## - this allows the new variable creation below

data = data.table(data)

## Create new variables

## - the code below adds a new column to the data (future\_pmpm)

## - the values for each row are set to allow\_future\_total/ member\_months\_future

data[, future\_pmpm:= allow\_future\_total/member\_months\_future]

data[, current\_pmpm:= allow\_current\_total/member\_months\_current]

## Remove outliers

data\_cut <- data[future\_pmpm<=36000]

data\_cut1 <- data\_cut[allow\_future\_total>1]

data\_cut2 <- data\_cut1[allow\_current\_total>1]

## Sample 10,000 observations

## - use the sample\_n() function from the dplyr library

data1 <- sample\_n(data\_cut2, 10000)

##================================================================

## Figure 8.1

##================================================================

## Figure 8.1 (a)

ggplot(data1, aes(x=future\_pmpm))+

geom\_histogram(aes(y = 100\*(..count..)/sum(..count..)), binwidth=1000, color='black', fill='grey')+

#scale\_y\_continuous(labels=percent) +

ylim(0,100) +

xlim(0,36000) +

xlab('Y($)') +

ylab('Percent') +

scale\_x\_continuous(breaks=pretty\_breaks(n=10))+

theme\_bw()

## Figure 8.1 (b)

qqnorm(y = data1$future\_pmpm,pch=8,cex=.6)

qqline( y = data1$future\_pmpm, col="black")

##================================================================

## Figure 8.2

##================================================================

## Figure 8.2 (a)

## Create log(Y) variables

## - syntax is the same as above

data1[, log\_future\_cost:=log(future\_pmpm)]

data1[, log\_current\_cost:=log(current\_pmpm)]

## Create plot

ggplot(data1, aes(x=log\_future\_cost))+

geom\_histogram(aes(y = (..density..)), binwidth=.3, color='black', fill='royalblue')+

scale\_y\_continuous(labels=percent)+

xlab('Log (Y)') +

ylab('Percent') +

scale\_x\_continuous(breaks=pretty\_breaks(n=50)) +

xlim(0,10.2)+

theme\_bw()+

stat\_function(fun = dnorm, args = list(mean = mean(data1$log\_future\_cost), sd = sd(data1$log\_future\_cost)),size=.75)

## Figure 8.2 (b)

qqnorm(y = data1$log\_future\_cost, col='black',pch=8,cex=.6)

qqline( y= data1$log\_future\_cost, col= 'royalblue')

##================================================================

## Figure 8.3

##================================================================

ggplot(data1, aes(x=current\_pmpm, y=future\_pmpm))+

geom\_point(pch=8,cex=.6) +

xlab('X($)')+

ylab('Y($)')+

theme\_bw()

##================================================================

## Figure 8.4

##================================================================

## Figure 8.4 (a)

ggplot(data1, aes(x=current\_pmpm, y=log\_future\_cost))+

geom\_point(pch=8,cex=.6)+

xlab('X1($)') +

ylab('Log(Y)')+

theme\_bw()

## Figure 8.4 (b)

ggplot(data1, aes(x=log\_current\_cost, y=log\_future\_cost))+

geom\_point(pch=8,cex=.6)+

xlab('Log(X1)') +

ylab('Log(Y)') +

scale\_x\_continuous(breaks=pretty\_breaks(n=50)) +

xlim(0,10)+

theme\_bw()

##================================================================

## Figure 8.4

##================================================================

## Create two new datasets based on gender

datam <- data1[gender=='M']

dataf <- data1[gender=='F']

## Figure 8.5 (a) - Female

ggplot(dataf, aes(x=log\_current\_cost, y=log\_future\_cost))+

geom\_point(pch=8,cex=.6)+

xlab('Log(X1)') +

ylab('Log(Y)')+

xlim(0,10)+

theme\_bw()

## Figure 8.5 (b) - Male

ggplot(datam, aes(x=log\_current\_cost, y=log\_future\_cost))+

geom\_point(pch=8,cex=.6)+

xlab('Log(X1)') +

ylab('Log(Y)')+

xlim(0,10)+

theme\_bw()

##================================================================

## Table 8.3

##================================================================

## Top half of the table

## Create new data table with variables of interest

correlation = data.table(data1$future\_pmpm, data1$current\_pmpm, data1$gender, keep.rownames = TRUE)

## Set column names

setnames(correlation, c('V1', 'V2', 'V3') ,c('future\_pmpm', 'current\_pmpm', 'gender'))

## Converting gender to a numeric variable (1 or 2 in this case) allows us to calculate correlation

correlation[, gender:=as.numeric(gender)]

## Calculate Correlation

cor(correlation)

## Bottom half of the table

## Create new data table with variables of interest (log variables)

log\_correlation = data.table(data1$log\_future\_cost, data1$log\_current\_cost, data1$gender, keep.rownames = TRUE)

## Set column Names

setnames(log\_correlation, c('V1', 'V2', 'V3') ,c('log\_future\_pmpm', 'log\_current\_pmpm', 'gender'))

## Converting gender to a numeric variable (1 or 2 in this case) allows us to calculate correlation

log\_correlation[, gender:=as.numeric(gender)]

## Calculate Correlation

cor(log\_correlation)

##================================================================

## Table 8.4

##================================================================

## Fit Regression model

model <- lm(future\_pmpm~., data=correlation)

## Table 8.4 (a)

summary(model)

## Table 8.4 (b)

anova(model)

##================================================================

## Table 8.5

##================================================================

## Fit Regression model

log\_model <- lm(log\_future\_pmpm~., data=log\_correlation)

## Table 8.5 (a)

summary(log\_model)

## Table 8.5 (b)

anova(log\_model)

##================================================================

## Table 8.6

##================================================================

## Calculate jack-knife residuals

jackknife = studres(model1)

model1 <- model

model1$residuals <- jackknife

## View residuals plots

## - The first and second plots corresponds to figures 8.6 (a) and (c), respectively

plot(model1,pch=8,cex=.6,col='black')

## Repeat process for the log-regression model

log\_jackknife <- studres(log\_model)

log\_model1 <- log\_model

log\_model1$residuals <- log\_jackknife

## View residuals plots

## - The first and second plots corresponds to figures 8.6 (b) and (d), respectively

plot(log\_model1,pch=8,cex=.6)

**Chapter 10**

**Appendix: R code for Logistic Regression Models**

## Install backing packages

## - note: do not run these lines if packages are already installed

install.packages("data.table")

install.packages("scales")

install.packages("dplyr")

install.packages('MASS')

install.packages('pROC')

## Load the libraries we just installed

library(data.table)

library(scales)

library(dplyr)

library(MASS)

library(pROC)

##===========================================================

## Data Pre-processing

##===========================================================

## Reading in the dataset

data <- read.csv("Blue\_book\_sample\_data\_updated.csv",

header=T)

## Convert data to a data.table object

data1 = data.table(data)

##===========================================================

## Section 10.3 Example of Logistic Regression

##===========================================================

## Table 10.3

admin\_table <- table(data1$admit\_flg\_current, data1$admit\_flg\_future)

## View table

admin\_table

## Chi-Squared Contingency table test

chisq.test(admin\_table)

## Logistic Regression Model

fit <- glm(admit\_flg\_future ~ admit\_flg\_current +gender+A\_OVER64+Er\_visit\_flg\_current+

pcp\_visit\_cnt\_current, family = binomial , data=data1)

## Table 10.4 (a)

summary(fit)

## Table 10.4 (b)

var.fit <- vcov(fit)

## View variance-covariance matrix

var.fit

##===========================================================

## Section 10.4 Predictive Accuracy

##===========================================================

## Use predict.glm() function and the model object (fit) to make new predictions

predictions <- predict.glm(fit, newdata= data1, type = "response")

## Add predictions to the data table

data1$predictions <- predictions

## Create binary prediction variable (0 or 1)

## Initially set the new variable (admit\_flg\_predict) to 0

data1$admit\_flg\_predict = 0

## If the value of a prediction is > .05 we set the admit flag = 1

## The following code is only possible using a data.table object

data1[predictions > .05, admit\_flg\_predict := 1]

## Table 10.5 / Table 10.6

## Create table using future admissions and predicted admissions

prediction.table <- table(data1$admit\_flg\_future, data1$admit\_flg\_predict)

## View prediction table

prediction.table

## Prediction Accuracy = (n00 + n11)/n

(80118+1656)/100000

# prediction accuracy = 81.77%

## Sensitivity = n11/(n10 + n11)

1656/(2998+1656)

## sensitivity = 35.58%

## Specificity = n00/(n00+n01)

80118/(15228+80118)

## specificity = 84.03%

##===========================================================

## Section 10.5 Optimal Cutoff Point

##===========================================================

## In order to re-produce figures 10.1 (a) and (b) we need to calculate prediction

## accuracy, sensitivity, and specificity as a function of our chosen cutoff point

## The user-defined function below does that:

cutoff\_values <- function(admit\_flg\_future, predictions, cutoff\_point){

# predictions - a vector of predicted readmission percentages (values range from 0:1)

# cutoff\_point - a single real number from (0:1)

admit\_flg\_predict <- rep(0,100000)

# If the prediction is greater than the specified cutoff point set

# admit\_flg\_predict = 1

for(i in 1:100000){

if(predictions[i] > cutoff\_point)

admit\_flg\_predict[i] = 1

}

# Count the number of correctly predicted non-admissions

count00 <- 0

for(i in 1:100000){

if(admit\_flg\_future[i] == 0 && admit\_flg\_predict[i] == 0)

count00 = count00 + 1

}

# Count the number of correctly predicted re-admissions

count11 <- 0

for(i in 1:100000){

if(admit\_flg\_future[i] == 1 && admit\_flg\_predict[i] == 1)

count11 = count11 + 1

}

# Count the number of incorrectly predicted non-admissions

count10 <- 0

for(i in 1:100000){

if(admit\_flg\_future[i] == 1 && admit\_flg\_predict[i] == 0)

count10 = count10 + 1

}

# Count the number of incorrectly predicted admissions

count01 <- 0

for(i in 1:100000){

if(admit\_flg\_future[i] == 0 && admit\_flg\_predict[i] == 1)

count01 = count01 + 1

}

prediction\_accuracy <- (count00 + count11)/100000

sensitivity = count11 / (count10 + count11)

specificity = count00 / (count01 + count00)

return(data.frame(prediction\_accuracy, sensitivity, specificity))

}

## We can now use this function to test the prediction accuracy of a cutoff point

## Example:

cutoff\_values(data1$admit\_flg\_future, data1$predictions, .05)

## This shows a cutoff = .05 has a prediction accuracy of 77.9%

## - this is consistent with our findings from table 10.5 and 10.6

## Test every possible cutoff point from .005 to .975

cutoffs <- seq(from =.005, to=.975, by = .005)

## Create an empty data frame to store results

test <- data.frame(prediction\_accuracy=0, sensitivity=0, specificity=0)

for(i in cutoffs){ #for loop will take some time depending on memory

test <- rbind(test, cutoff\_values(data1$admit\_flg\_future, data1$predictions, i))

}

##===============================

# Figure 10.1 (a)

##===============================

plot(cutoffs, test$prediction\_accuracy[2:196], xlab = "Cutoff Points", ylab = "Predictive Accuracy", ylim=c(0,1), cex=.4, pch=2)

## Draw a smooth line through points

lines(loess(test$prediction\_accuracy[2:196]~cutoffs), lwd=3)

## Plot max prediction accuracy

abline(h = max(test$prediction\_accuracy), lty=2)

## Plot maximum cutoff line

abline(v=max(data1$predictions), lty=3)

## Create legend

legend(x = .3, y = .7, c("Predictive Accuracy", "Prediction Accuracy = .955", "Cutoff = .9739"), cex =.8, lty = c(1,2,3), bty = 'n')

##===============================

# Figure 10.1 (b)

##===============================

plot(cutoffs, test$specificity[2:196], type = "l", lwd=3, xlab = "Cutoff Points",

ylab = "Predictive Accuracy")

## Add specificity line

lines(cutoffs, test$sensitivity[2:196], type = "l", lwd=1, lty=2)

## Add curb (intersection between sensitivity and specificity)

## To find this value we can look at where sensitivity > specificity

## - look for when FALSE changes to TRUE --> i.e. the lines intersected

## - this corresponds to the 10th observation (or 9th cutoff point)

test$specificity > test$sensitivity

abline(v = cutoffs[9], lty=3)

## Create Legend

legend(x = .3, y = .7, c("Specificity", "Sensitivity", "Curb = .045"), lty = c(1,2,3),

bty = 'n', cex = .8)

##===========================================================

## Section 10.6 ROC Curve

##===========================================================

## Figure 10.2 - ROC Curve

roc.curve <- roc(data1$admit\_flg\_future, data1$predictions, plot = T)

## Plot curve

plot(roc.curve, main= "ROC Curve for Model")

mtext("Area under the Curve = .6372")

##===========================================================

## 10.6.1 Lift and Percentage of Captured Events

##===========================================================

## First order data by descending prediction % to create order for quintiles

ordered.data1 <- data1[order(predictions, decreasing=TRUE),]

ordered.data <- data.frame(ordered.data1)

## Table 10.8

## Calculate actual number of admissions by quintile

actual\_admissions <- c(sum(ordered.data$admit\_flg\_future[1:20000]),

sum(ordered.data$admit\_flg\_future[20001:40000]),

sum(ordered.data$admit\_flg\_future[40001:60000]),

sum(ordered.data$admit\_flg\_future[60001:80000]),

sum(ordered.data$admit\_flg\_future[80001:100000]))

## Calculate expected admissions per quintile

expected\_admissions <- c(sum(ordered.data1$predictions[1:20000]),

sum(ordered.data1$predictions[20001:40000]),

sum(ordered.data1$predictions[40001:60000]),

sum(ordered.data1$predictions[60001:80000]),

sum(ordered.data1$predictions[80001:100000]))

## Create table 10.8

table10.8 <- data.frame(quintiles = c(1:5),

expected\_admissions,

actual\_admissions,

adults\_in\_quintile <- rep(20000,5),

percent\_admissions\_in\_quintile = actual\_admissions/4654,

lift = actual\_admissions/(4654/5))

## View table

View(table10.8)

## Table 10.9

## Calculate cumulative life

actual\_admissions2 <- actual\_admissions[5:1]

expected\_admissions2 <- expected\_admissions[5:1]

cumulative\_lift <- rep(0,5)

for(i in 1:5){

cumulative\_lift[i] <- actual\_admissions2[i]/(930.8\*i)

actual\_admissions2[i+1] = actual\_admissions2[i] + actual\_admissions2[i+1]

expected\_admissions2[i+1] = expected\_admissions2[i] + expected\_admissions2[i+1]

}

## Create Table 10.9

table10.9 <- data.frame(quintiles = c(1:5),

expected\_admissions2[1:5],

actual\_admissions2[1:5],

adults\_in\_quintile <- rep(20000,5),

actual\_admissions2[1:5]/4654,

cumulative\_lift)

## View table

View(table10.9)

## Plot results

plot(table10.8$quintiles, table10.8$lift, col = "blue",ch = 18, type = "o",

xlab = "", ylab = "", main = "Lift Charts")

lines(table10.9$quintiles, table10.9$cumulative\_lift, col = "red", pch = 15, type = "o")

legend(x = 3.5, y = 2, c("Discrete Lift", "Cumulative Lift"), col = c("blue", "red"), pch = c(18,15) , bty = 'n')

**Chapter 11**

**Appendix: R Code**

library("tree")

library("caret")

library("ROCR")

library("randomForest")

maindat<- read.csv("dataChapter11Bluebook.csv",header=TRUE)

dim(maindat)

str(maindat)

## split data into training set and test set:

set.seed(89)

trainIndex <- createDataPartition(maindat$bp, p=.7,list=FALSE)

train.set<- maindat[trainIndex,] # 70% of the whole data

test.set<- maindat[-trainIndex,] # 30% of the whole data

################ Decision Tree ##################

## Build the tree on the train set:

fit<- tree(bp~.,data=train.set,method="class")

plot(fit)

text(fit,pretty=0)

## Predict on the test set:

fit.predict<- predict(fit,test.set,type="class")

## Confusion matrix:

error.matrix<- table(fit.predict,test.set$bp)

## Classification error:

1-sum(diag(error.matrix))/sum(error.matrix)

## Tree pruning (avoid overfitting):

set.seed(2879)

# k-fold cross validation:

cv.fit<- cv.tree(fit,FUN=prune.tree,K=10,method="misclass")

par(mfrow=c(1,2))

plot(cv.fit$size,cv.fit$dev,xlab="Number of nodes",ylab="CV Misclassification Error",type = "b",lwd=2,col="red")

plot(cv.fit$k,cv.fit$dev,xlab="Complexity Parameter",type="b",lwd=2,col="blue")

prune.fit<- prune.misclass(fit,best=4)

par(mfrow=c(1,1))

plot(prune.fit)

text(prune.fit,pretty=0,cex =1)

# Prediction using prune tree:

predict.prune<- predict(prune.fit,test.set,type="class")

error<-table(predict.prune,test.set$bp)

1-sum(diag(error))/sum(error)

# auc value:

tree.pred.prob <- predict(prune.fit,test.set,type=c("vector"))

pred <- prediction(tree.pred.prob[,2],test.set$bp)

perf <- performance(pred, measure = "tpr", x.measure = "fpr")

plot(perf, col=rainbow(7), main="ROC curve", xlab="Specificity",

ylab="Sensitivity",lwd=4)

abline(0, 1) #add a 45 degree line

auc.tmp<- performance(pred,"auc")

(auc<- as.numeric(auc.tmp@y.values)) # c-statistis is .73

################## Bagging ##############

set.seed(1)

# bagged model:

bag.tree<- randomForest(bp~.,data=train.set,mtry=10,importance=TRUE,ntree=1000)

# important variables:

importance(bag.tree)

varImpPlot(bag.tree,type=2)

yhat.bag<- predict(bag.tree,test.set,type="class")

## prediction error:

bag.error<- table(yhat.bag,test.set$bp)

(1-sum(diag(bag.error))/sum(bag.error))

# auc value:

bag.pred.prob<- predict(bag.tree,test.set,type="prob")

bag.pred<- prediction(bag.pred.prob[,2],test.set$bp)

perf <- performance(bag.pred,measure = "tpr", x.measure = "fpr")

plot(perf, col=rainbow(7), main="ROC curve", xlab="Specificity",

ylab="Sensitivity",lwd=4)

abline(0, 1) #add a 45 degree line

auc.tmp<- performance(bag.pred,"auc")

(auc<- as.numeric(auc.tmp@y.values)) # 0.92

############# Random Forest ################

set.seed(23)

# random forest:

rf.tree<- randomForest(bp~.,data=train.set,mtry=sqrt(10),importance=TRUE,ntree=1000)

importance(rf.tree)

varImpPlot(rf.tree,type=2)

# predict rf.tree on test set:

yhat.rf<- predict(rf.tree,newdata =test.set,type="class")

error.rf<- table(yhat.rf,test.set$bp)

# create roc curve:

rf.pred.prob<- predict(rf.tree,test.set,type=c("prob"))

rf.pred <- prediction(rf.pred.prob[,2],test.set$bp)

perf <- performance(rf.pred,measure = "tpr", x.measure = "fpr")

plot(perf, col=rainbow(7), main="ROC curve", xlab="Specificity",

ylab="Sensitivity",lwd=4)

abline(0, 1) #add a 45 degree line

auc.tmp<- performance(rf.pred,"auc")

(auc<- as.numeric(auc.tmp@y.values)) # 0.93

**Chapter 20**

**Appendix: R Code for Running the Model**

# import the data set

setwd("~/Dropbox/196 package")

library("data.table")

data<- as.data.table(read.csv("Dataset196.csv",header=T))

# 1.DRG Classification --------------------------------------------

# Classify DRG into 2 groups: DRG medical/surgical.Link of classification is below:

# Https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/MedicareFeeforSvcPartsAB/downloads/DRGdesc08.pdf

# valid codes for DRG surgical

surgicalG<- c(1:42,113:117,129:139,163:168,215:264,326:358,405:425,453:517,573:585,614:630,652:675,707:718,734:750,765:770,799:804,820:830,853:858,876,901:909,927:929,939:941,955:959,969:970,981:989)

# valid codes for DRG medical

medicalG<- c(52:103,121:125,146:159,175:208,280:316,368:395,432:446,533:566,592:607,637:645,682:700,722:730,754:761,774:795,808:816,834:849,862:872,880:897,913:923,933:935,945:951,963:965,974:977)

# create DRG.Class variable

data[DRG %in% surgicalG,DRG.Class:=as.factor("SURG")]

data[DRG %in% medicalG,DRG.Class:=as.factor("MED")]

data[is.na(DRG.Class),DRG.Class:=as.factor("UNGROUP")]

# 2.Determine length of stay --------------------------------------data[,LOS:=as.Date(as.character(Discharge.Date),'%m/%d/%Y')- as.Date(as.character(Admit.Date),"%m/%d/%Y") +1]

# 3.Determine age (up to the date of admission) -------------------

data[,Age:=year(as.Date(as.character(Admit.Date),"%m/%d/%y"))-year(as.Date(as.character(Birthday),'%Y%m%d'))]

# 4.Changing labels for Gender ------------------------------------

data[,Gender:=as.factor(Gender)]

levels(data$Gender)<- c("M","F")

# 5. Changing labels for Race -------------------------------------

data[,Race:=as.factor(Race)]

levels(data$Race)<- c("White","Black","Others","Hispanic")

# 6.Calculate HCC riskscore ---------------------------------------

# A. Calculate demographic riskscore

# a) Subset important information to calculate riskscore: need age,gender,and HCCs

demo.get<- data[,.(ID.Codes,Age,Gender)]

# b) create a data table that contains risk-score for each demographic group

Age<-rep(seq(0,120),2)

Gender<- c(rep("F",121),rep("M",121))

demo.score<- c(rep(0.198,35),rep(0.212,10),rep(.274,10),rep(.359,5),rep(.416,5),rep(.283,5),rep(.346,5),rep(.428,5),rep(.517,5),rep(.632,5),rep(.755,5),rep(.775,26),rep(.079,35),rep(.119,10),rep(.165,10),rep(.292,5),rep(.332,5),rep(.309,5),rep(.378,5),rep(.464,5),rep(.565,5),rep(.647,5),rep(.776,5),rep(.963,26))

demo<- as.data.table(cbind(Age,Gender,demo.score))

# convert age and demo.score into numerics:

cols<- c("Age","demo.score")

demo[,(cols):=lapply(.SD,as.numeric),.SDcols=cols]

# convert Gender into categorical:

demo[,Gender:=as.factor(Gender)]

# c) Merge demo into demo.get, using Age and Gender columns to merge:

demo.get<- merge(demo.get,demo,by=c("Age","Gender"))

demo.get<- demo.get[order(ID.Codes)]

rm(demo)

# B. Calculate disease riskscore:

# a) Subset important information to calculate riskscore:

hcc.get<- data[,c(22:100),with=FALSE]

hcc.get<- matrix(sapply(hcc.get,as.numeric),nrow=66782,ncol=79)

# b) Input Disease Coefficients (Community Factor):

# Use data in Table 1:Preliminary Community and Institutional Relative Factors for the CMS-HCC Risk Adjustment Model

# this data set doesn't have HCC51, HCC52, HCC138, HCC139, HCC140, HCC141, HCC159, HCC160

diseaseC<- as.matrix(c(.492,.520,.557,2.425,1.006,0.695,.330,.180,0.334,.334,.124,.653,.342,.240,1.003,.425,.313,.337,.257,.279,.423,.376,1.078,.306,.258,.358,.358,.471,.318,1.075,.868,.441,1.016,.036,.281,.460,.482,.555,.252,.533,1.732,.769,.326,.361,.283,.283,.210,.276,.371,.333,.481,.212,1.313,.417,.288,.388,.388,.294,.691,.212,.223,.248,.617,.617,.227,.277,1.071,1.071,.473,.458,.533,.141,.441,.363,.379,.555,1.032,.609,0.804),nrow=79,ncol=1)

# c) the riskscore vector is the multiplication between hcc.get and diseaseC:

hcc.get<- as.data.table(hcc.get %\*% diseaseC)

hcc.get<- cbind(data$ID.Codes,hcc.get)

names(hcc.get)<- c("ID.Codes","hcc.score")

hcc.get<- hcc.get[order(ID.Codes)]

# C. Calculate the total HCC riskscore and add it into the big data set:

data<- data[order(ID.Codes)]

data$HCC.Riskscore<- demo.get$demo.score+hcc.get$hcc.score

# 7. Mapping DRG Complication ------------------------------------SurgMCC.CC<-c(1,5,11,20,23,25,28,31,34,37,40,163,166,216,219,222,224,226,228,231,233,235,237,239,242,246,248,250,252,255,258,260,326,329,332,335,338,341,347,420,423,453,456,459,461,463,466,469,471,474,477,480,485,492,495,500,503,510,515,573,576,579,616,619,622,625,628,653,656,659,662,665,668,673,736,739,799,802,820, 823,826,856,901,907,939,957,969,981,984,987,12,21,26,29,32,35,38,41,113,116,129,131,133,135,137,164,167,217,220,229,240,243,253,256,261,327,330,333,336,339,342,345,348,351,354,357,464,467,472,475,478,481,483,486,488,490,493,496,498,501,504,507,511,513,516,574,577,580,582,584,614,617,620,623,626,629,654,657,660,663,666,669,671,674,707,709,711,713,715,717,734,737,740,742,744,746,749,800,803,821,824,827,829,854,857,902,908,928,940,958,982,985,988)

SurgNoC<- c(2,6,13,22,24,27,30,33,36,39,42,114,117,130,132,134,136,138,165,168,218,219,220,221,223,224,225,226,227,230,232,234:236,238,241,244,247,249,250,251,254,257,259,262,328,331,334,337,340:343,349,352,355,358,407:410,413:419,422,425,455,459,460,462,465,468,470,473,476,479,482,484,487,489,491,494,497,499,502,505,508,512,514,517,575,578,581,583,661,664,667,670,672,675,708,710,712,714,716,718,735,738,741,743,745,747,750,766,801,804,822,825,828,830,855,858,903,905,909,929,941,959,970,983,986,989)

MedicalMCC.CC<- c(54,56,58,61,64,67,70,73,77,80,82,85,88,91,94,97,100,102,124,146,150,152,154,157,175,177,180,183,186,190,193,196,199,205,280,283,286,288,291,296,299,302,304,306,308,314,368,371,374,377,380,383,385,388,391,393,432,435,438,441,444,533,535,539,542,545,548,551,553,555,557,559,562,564,592,595,597,602,604,606,637,640,643,682,686,689,693,698,722,725,727,754,757,808,811,814,834,837,840,843,846,862,865,867,871,896,913,915,917,922,947,963,974,52,59,62,65,71,75,78,83,86,89,92,95,98,121,147,155,158,178,181,184,187,191,194,197,200,202,281,284,289,292,294,297,300,309,315,369,372,375,378,381,386,389,394,433,436,439,442,537,540,543,546,549,560,565,593,598,600,638,644,687,691,699,723,729,755,758,760,765,809,815,835,838,841,844,847,868,920,945,949,964,975)

MedicalNoC<- c(53,55,60,63,66,67,68,72,74,76,79,81,84,87,90,96,99,101,103,122,125,148,151,156,159,176,179,182,185,188,192,195,198,201,203,206,282,285,287,290,293,295,298,301,303,305,307,310,316,370,373,376,379,382,384,387,390,392,395,434,437,440,443,446,534,536,538,541,544,547,550,552,554,556,558,561,563,566,594,596,599,601,603,605,607,639,641,645,684,688,690,692,693,694,696,700,724,728,730,756,759,761,775,810,812,816,834,835,836,839,842,845:848,866,869,871,872,897,914,916,918,921,923,933,946,948,950,965,976,977)

data[DRG %in% SurgMCC.CC, DRG.Complication:=as.factor("SurgMCC.CC")]

data[DRG %in% SurgNoC, DRG.Complication:=as.factor("SurgNoC")]

data[DRG %in% MedicalMCC.CC, DRG.Complication:=as.factor("MedicalMCC.CC")]

data[DRG %in% MedicalNoC, DRG.Complication:=as.factor("MedicalNoC")]

data[is.na(DRG.Complication),DRG.Complication:=as.factor("Other")]

# 8. Budilding Logistic Regression --------------------------------

# subset important variables to build the model

final<- data[,c(1,4,7:8,101:106)]

# make sure the response variable is categorical

final$Readmission.Status<-as.factor(final$Readmission.Status)

# split data into training and test set, using training set to build model and test set to validate the model

set.seed(1)

# 70% of data as training set

train <- sample(1:nrow(final),46747)

final.train<-final[train,]

final.test<- final[-train,]

# proportional binomial/logit model:

fit.train<-glm(Readmission.Status~ Age + Gender + LOS + HCC.Riskscore + Race + DRG.Class + ER,family="binomial",data=final.train)

summary(fit.train)

# this model is built on the training set

# Variable selection and model selection:

# stepwise regression using backward elimination method (without any interaction terms)

fit2<- step(fit.train,direction="backward")

# investigate interaction terms in the model

library("MASS")

fit3<- update(fit2,.~.^2)

summary(fit3)

# Using chi-square test to perform deviance analysis:

anova(fit2,fit3,test="Chi") #--> prefer the reduced model: fit2

anova(fit2,fit.train,test="Chi") #--> prefer the reduced model: fit3

# fit this model on the test set:

fitpreds = predict(fit2,newdata=final.test,type="response")

# 9. Cutoff value and related plots -------------------------------

# determine the optimal cutoff value (where sensitivity==specificity):

library("ROCR")

fitpredsk<- prediction(fitpreds,final.test$Readmission.Status)

t<- performance(fitpredsk,"ppv")

k<-unlist(t@x.values)

k2<-unlist(t@y.values)

y<- as.numeric(final.test$Readmission)-1

perf = function(cut, fitpreds,y)

{

yhat = (fitpreds>cut) ## logical value: TRUE or FALSE if predicted prob. >cutoff

w = which(y==1) #index of true population of readmission cases

sensitivity = mean( yhat[w] == 1 ) # probability of readmission given that the patient is readmitted

specificity = mean( yhat[-w] == 0 ) # probability of no readmission given that the patient is not readmitted

c.rate = mean( y==yhat )

d = cbind(sensitivity,specificity)-c(1,1)

d = sqrt( d[1]^2 + d[2]^2 )

out = t(as.matrix(c(sensitivity, specificity, c.rate,d)))

colnames(out) = c("sensitivity", "specificity", "c.rate", "distance")

return(out)

}

s = seq(.001,.99,length=1000)

OUT = matrix(0,1000,4)

for(i in 1:1000) OUT[i,]=perf(s[i],fitpreds,y)

plot(s,OUT[,1],xlab="Cutoff",ylab="Value",cex.lab=1.5,cex.axis=1.5,ylim=c(0,1),type="l",lwd=8,axes=FALSE,col=2,

main="Fit 2")

axis(1,seq(0,1,length=5),seq(0,1,length=5),cex.lab=2)

axis(2,seq(0,1,length=5),seq(0,1,length=5),cex.lab=2)

lines(s,OUT[,2],col="darkgreen",lwd=8)

# lines(k,k2,lwd=2,col="black")

box()

legend(.25,.8,col=c(2,"darkgreen"),cex=1,lwd=c(3,3,3,3),c("Sensitivity","Specificity"))

abline(v=0.11,lty=3,lwd=2)

abline(0,1,lty=2)

points(.11,0.6638,pch=19,lwd=10)

## The intersection between sensitivity and specificity curves is 0.11

# obtain ROC curve for this model:

plot(1-OUT[,2],OUT[,1],main="ROC Curve",

xlab=c("1-Specificity"), ylab="Sensitivity",

type="l",lwd=10,col="orange")

abline(0,1)

# obtain c-statistic or area under the curve:

(c.stat<- performance(fitpredsk,measure="auc")@y.values)

# 10. Model performance by quantiles ------------------------------# Find the quantiles and the mean of prediction within each quantile:

quan<- quantile(fitpreds,c(0,.1,.2,.3,.4,.5,.6,.7,.8,.9,1))

# mean prediction within each quantile:

mean<- c()

for (i in 2:11){

mean[i-1]<- mean(fitpreds[(fitpreds>= quan[i-1])&(fitpreds<= quan[i])])

}

# actual cases of readmission within each quantiles:

actualOut<- c()

for (i in 2:11){

actualOut[i-1]<- length(which((fitpreds>= quan[i-1]) & (fitpreds<= quan[i]) & actual==1))

}

# number of observations in each quantile:

num<- c()

for (i in 2:11){

num[i-1]<- length(fitpreds[(fitpreds>= quan[i-1])&(fitpreds<= quan[i])])

}

# predicted outcomes:

predictedOut<- mean\*num

## Model Performance by Quantiles Plot:

actual<- c(32,62,104,136,189,228,348,335,466,649)

predicted<- c(103,118,131,146,165,192,230,292,406,745)

plot(actual,type="l",lwd=6,col="orange",

xlim=c(0,10),xaxt = "n",xlab="Quantiles",

ylim=c(0,800),ylab="Number of outcomes",

cex.lab=1)

grid()

points(predicted,type="l",lwd=6,col="darkturquoise")

axis(1, at=1:10, labels=c(0.1,.2,.3,.4,.5,.6,.7,.8,.9,1),lwd=4)

axis(2,lwd=4)

title("Model Performance by Quantiles")

box()

legend(1,400,col=c("orange","darkturquoise"),cex=1,lwd=c(2,2,2,2),c("Actual","Predicted"))