DS 740 Final Project

Connie Sosa

December 16, 2016

## R Markdown

diabetes.csv

##################################################################  
### DS 740: Final Project  
### Connie Sosa: 12/05/2016  
### Dataset from kaggle.com  
### diabetes.csv  
  
setwd("C:/Users/connie/UW/DS740-DataMining/week13/dataSets")  
diabetes = read.csv("diabetes.csv")  
  
### https://cran.r-project.org/web/packages/stargazer/vignettes/stargazer.pdf  
if (!require(stargazer)) {  
 install.packages("stargazer", repos="http://cran.rstudio.com") }

## Loading required package: stargazer

## Warning: package 'stargazer' was built under R version 3.3.2

##   
## Please cite as:

## Hlavac, Marek (2015). stargazer: Well-Formatted Regression and Summary Statistics Tables.

## R package version 5.2. http://CRAN.R-project.org/package=stargazer

if (!require(e1071)) {  
 install.packages("e1071", repos="http://cran.rstudio.com") }

## Loading required package: e1071

if (!require(pROC)) {  
 install.packages("pROC", repos="http://cran.rstudio.com")  
}

## Loading required package: pROC

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

##   
## Attaching package: 'pROC'

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

library(stargazer)  
library(e1071) ### SVM  
library(pROC) ### roc curves  
  
stargazer(diabetes, type="text")

##   
## ==========================================================  
## Statistic N Mean St. Dev. Min Max   
## ----------------------------------------------------------  
## Pregnancies 768 3.845 3.370 0 17   
## Glucose 768 120.895 31.973 0 199   
## BloodPressure 768 69.105 19.356 0 122   
## SkinThickness 768 20.536 15.952 0 99   
## Insulin 768 79.799 115.244 0 846   
## BMI 768 31.993 7.884 0.000 67.100  
## DiabetesPedigreeFunction 768 0.472 0.331 0.078 2.420   
## Age 768 33.241 11.760 21 81   
## Outcome 768 0.349 0.477 0 1   
## ----------------------------------------------------------

###########################################################  
### Imputation:  
### Columns Glucose,BloodPressure,and BMI should not have zero value.  
### Impute zero value in columns 2,3,6 with average value for each of the column.  
i = 1  
nCnt = dim(diabetes)[1]  
zeroValueCnt = rep(0, 3)  
NameIndexOfzeroValue = c(2,3,6)  
for(j in NameIndexOfzeroValue) {  
 zeroValueInd = which(diabetes[, j] == 0)  
 zeroValueCnt[i] = length(zeroValueInd)  
 diabetes[zeroValueInd, j] = mean(diabetes[, j])  
 i = i + 1  
}  
zeroValueM = matrix(c(names(diabetes)[NameIndexOfzeroValue],   
 zeroValueCnt,   
 round(zeroValueCnt/nCnt,4)), nrow=3, byrow = TRUE)  
stargazer(diabetes, type="text")

##   
## ============================================================  
## Statistic N Mean St. Dev. Min Max   
## ------------------------------------------------------------  
## Pregnancies 768 3.845 3.370 0 17   
## Glucose 768 121.682 30.436 44.000 199.000  
## BloodPressure 768 72.255 12.116 24.000 122.000  
## SkinThickness 768 20.536 15.952 0 99   
## Insulin 768 79.799 115.244 0 846   
## BMI 768 32.451 6.875 18.200 67.100   
## DiabetesPedigreeFunction 768 0.472 0.331 0.078 2.420   
## Age 768 33.241 11.760 21 81   
## Outcome 768 0.349 0.477 0 1   
## ------------------------------------------------------------

diabetes$Outcome = as.factor(diabetes$Outcome)  
  
#############################################################################  
#### Double cross-validation:  
#### Two models:   
nmodels = 2 ## Model 1 - Logistic Regression, Model 2 - Support Vector Machine (SVM)  
  
##### model assessment OUTER CV (with model selection INNER CV as part of model-fitting) #####  
fulldata.out = diabetes  
k.out = 10 ### use 10 fold CV  
n.out = dim(fulldata.out)[1]  
#### define the cross-validation splits for assessment  
numPerFold= floor(n.out/k.out)  
groups.out = c(rep(1:k.out, numPerFold), 1:(n.out%%k.out)) ### produces list of group labels  
# groups.out = c(rep(1:k.out,floor(n.out/k.out)),1:(n.out-floor(n.out/k.out)\*k.out))   
  
set.seed(14) ### For week 14, use seed 14  
cvgroups.out = sample(groups.out,n.out)   
allpredictedCV1.out = rep(NA, n.out)  
allpredictedCV2.out = rep(NA, n.out)  
for (j in 1:k.out) {   
 bestmodel.in = 0  
 groupj.out = (cvgroups.out == j)   
 traindata.out = diabetes[!groupj.out,]  
 testdata.out = diabetes[groupj.out,]  
   
 ####################################  
 ### entire model-fitting process ###  
 ####################################  
 fulldata.in = traindata.out # only input the data used to fit the model  
 k.in = 10   
 n.in = dim(fulldata.in)[1]  
 groups.in = c(rep(1:k.in, floor(n.in/k.in)), 1:(n.in%%k.in))   
 cvgroups.in = sample(groups.in, n.in)   
   
 predictLogistic.in = rep(-1, n.in)  
 predictSVM.in = rep(0, n.in)  
 for (i in 1:k.in) {  
 groupi.in = (cvgroups.in == i)  
 ###############################  
 ### model 1: Logistic Regression   
 ###############################  
 logistfit.in = glm(Outcome~., data=diabetes[!groupi.in,], family="binomial")  
 predictLogistic.in[groupi.in]=predict(logistfit.in, fulldata.in[groupi.in,],type="response")  
  
 ###############################   
 ### model 2: Support Vector Machine  
 ###############################  
 ### Use cross validation for SVM model selection:  
 SVMfit.in = tune(svm, Outcome~., data=diabetes[!groupi.in, ], kernel = "radial",  
 ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100),  
 gamma = c(0.5, 1, 2, 3, 4)), type="C-classification")  
 predictSVM.in[groupi.in] = attributes(predict(SVMfit.in$best.model, newdata=fulldata.in[groupi.in,], decision.values=TRUE))$decision.values  
 bestSVMCost = summary(SVMfit.in)$best.model$cost  
 bestSVMGamma = summary(SVMfit.in)$best.model$gamma  
 } ### end of INNER loop i  
   
 ### Model 1: Logistic Regression: confusion matrix  
 confuseMatrix = matrix(0, nr=2, nc=2)  
 confuseMatrix = table(predictLogistic.in > 0.5, fulldata.in[cvgroups.in,]$Outcome)  
  
 ### Model 2: SVM: confusion matrix   
 confuseMatrix2 = matrix(0, nr=2, nc=2)  
 confuseMatrix2 = table(predictSVM.in > 0.5, fulldata.in[cvgroups.in,]$Outcome)  
   
 ### Classification Error Rate:  
 ### take the sum of off diagonal confusion matrix and divide it by n.in  
 allmodelCV.in = rep(NA, nmodels)   
 allmodelCV.in[1] = (confuseMatrix[1,2] + confuseMatrix[2,1])/n.in   
 allmodelCV.in[2] = (confuseMatrix2[1,2] + confuseMatrix2[2,1])/n.in   
   
 ################################ ###  
 ### resulting in bestmodel.in ###  
 ################################ ###  
 ### Saving both models' predictions for plotting ROC curve and comparison.  
   
 #############################  
 ### Save logistic model's prediction:  
 #############################  
 fitCV1.out = glm(Outcome~., data=traindata.out, family="binomial")  
 allpredictedCV1.out[groupj.out]=predict(fitCV1.out, testdata.out, type="response")  
  
 #############################  
 ### Save SVM model's prediction:  
 #############################  
 ### FIT the best radial SVM model on training data  
 svmfitRadial = svm(Outcome~., data =traindata.out, kernel = "radial",   
 cost = summary(SVMfit.in)$best.model$cost, gamma = summary(SVMfit.in)$best.model$gamma,  
 decision.values=T, type="C-classification")  
 ### PREDICT on test data  
 xgrid = data.frame(testdata.out)  
 allpredictedCV2.out[groupj.out] = attributes(predict(svmfitRadial, xgrid, decision.values = TRUE))$decision.values  
  
 if (allmodelCV.in[1] <= allmodelCV.in[2]) {  
 bestmodel.in = 1 ### Best model is logistic regression  
 } else {  
 bestmodel.in = 2 ### Best model is SVM  
 } ### end of else  
} ### end of OUTER loop j  
##################################################################  
### expensive grid search, takes a couple of hours to run.  
  
###########################  
### assessment  
###########################  
### ROC curves compare logistic regression and SVM, p365 reference  
  
#############  
### ROC curve: for logistic regression  
#############  
table(allpredictedCV1.out > 0.5, fulldata.out$Outcome)

##   
## 0 1  
## FALSE 444 116  
## TRUE 56 152

myrocLogistic = roc(response=fulldata.out$Outcome, predictor=allpredictedCV1.out)  
plot.roc(myrocLogistic, col="navy", lwd=1, lty=1, main="ROC Curves")

##   
## Call:  
## roc.default(response = fulldata.out$Outcome, predictor = allpredictedCV1.out)  
##   
## Data: allpredictedCV1.out in 500 controls (fulldata.out$Outcome 0) < 268 cases (fulldata.out$Outcome 1).  
## Area under the curve: 0.8367

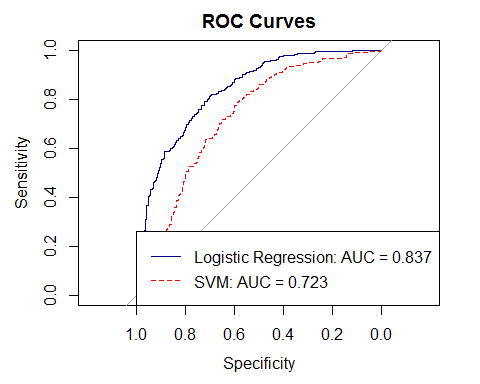
aucLogi = paste("Logistic Regression: AUC = ", round(myrocLogistic$auc,3), sep="")  
  
#############  
### ROC curves: for SVM  
#############  
table(allpredictedCV2.out > 0.5, fulldata.out$Outcome)

##   
## 0 1  
## FALSE 447 211  
## TRUE 53 57

myrocSVM = roc(response=fulldata.out$Outcome, predictor=allpredictedCV2.out)  
aucSVM = paste("SVM: AUC = ", round(myrocSVM$auc,3), sep="")  
plot.roc(myrocSVM, col="red", lwd=1, lty=2, add=TRUE)

##   
## Call:  
## roc.default(response = fulldata.out$Outcome, predictor = allpredictedCV2.out)  
##   
## Data: allpredictedCV2.out in 500 controls (fulldata.out$Outcome 0) < 268 cases (fulldata.out$Outcome 1).  
## Area under the curve: 0.723

legend("bottomright", c(aucLogi,aucSVM), lty=c(1,2), lwd=1, col=c("navy", "red"))



###########################  
### Fit the selected model to the entire dataset  
###########################  
if (bestmodel.in == 1) {  
 ### best mode is logistic regression  
 fitbestModel = glm(Outcome~., data = diabetes, family="binomial")  
} else {  
 ### best mode is SVM  
 fitbestModel = svm(Outcome~., data = diabetes, kernel = "radial",   
 cost = bestSVMCost, gamma = bestSVMGamma, decision.values=T, type="C-classification")  
}  
fitbestModel

##   
## Call: glm(formula = Outcome ~ ., family = "binomial", data = diabetes)  
##   
## Coefficients:  
## (Intercept) Pregnancies   
## -9.166163 0.124045   
## Glucose BloodPressure   
## 0.038624 -0.011587   
## SkinThickness Insulin   
## -0.003029 -0.001380   
## BMI DiabetesPedigreeFunction   
## 0.101655 0.940634   
## Age   
## 0.011872   
##   
## Degrees of Freedom: 767 Total (i.e. Null); 759 Residual  
## Null Deviance: 993.5   
## Residual Deviance: 709.8 AIC: 727.8

summary(fitbestModel)

##   
## Call:  
## glm(formula = Outcome ~ ., family = "binomial", data = diabetes)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.6981 -0.7199 -0.3982 0.7110 2.4249   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -9.1661632 0.8161021 -11.232 < 2e-16 \*\*\*  
## Pregnancies 0.1240454 0.0324399 3.824 0.000131 \*\*\*  
## Glucose 0.0386243 0.0039000 9.904 < 2e-16 \*\*\*  
## BloodPressure -0.0115866 0.0086364 -1.342 0.179725   
## SkinThickness -0.0030286 0.0068301 -0.443 0.657465   
## Insulin -0.0013801 0.0009197 -1.501 0.133476   
## BMI 0.1016546 0.0165237 6.152 7.65e-10 \*\*\*  
## DiabetesPedigreeFunction 0.9406336 0.3015953 3.119 0.001816 \*\*   
## Age 0.0118721 0.0095185 1.247 0.212297   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 993.48 on 767 degrees of freedom  
## Residual deviance: 709.81 on 759 degrees of freedom  
## AIC: 727.81  
##   
## Number of Fisher Scoring iterations: 5