CW6

HT

6/24/2019

Note that the echo = FALSE parameter was added to the code chunk to prevent printing of the R code that generated the plot.

# Importing libraries

library(dplyr) # For mutating columns

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(ggplot2)# To generate Plots  
library(scales) # Converting to %  
library(kknn) # For model quality and accuracy  
  
rm(list = ls())

# Importing and exploring data

set.seed(562)  
  
# Importing Data  
bc.data <- read.delim("breast-cancer-wisconsin.data.txt", stringsAsFactors = FALSE, header = FALSE, sep = ',')  
  
head(bc.data)

## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11  
## 1 1000025 5 1 1 1 2 1 3 1 1 2  
## 2 1002945 5 4 4 5 7 10 3 2 1 2  
## 3 1015425 3 1 1 1 2 2 3 1 1 2  
## 4 1016277 6 8 8 1 3 4 3 7 1 2  
## 5 1017023 4 1 1 3 2 1 3 1 1 2  
## 6 1017122 8 10 10 8 7 10 9 7 1 4

# Function to identify the missing values  
  
for (i in 2:11) {  
 print(paste0("V",i))  
 print(table(bc.data[,i]))  
}

## [1] "V2"  
##   
## 1 2 3 4 5 6 7 8 9 10   
## 145 50 108 80 130 34 23 46 14 69   
## [1] "V3"  
##   
## 1 2 3 4 5 6 7 8 9 10   
## 384 45 52 40 30 27 19 29 6 67   
## [1] "V4"  
##   
## 1 2 3 4 5 6 7 8 9 10   
## 353 59 56 44 34 30 30 28 7 58   
## [1] "V5"  
##   
## 1 2 3 4 5 6 7 8 9 10   
## 407 58 58 33 23 22 13 25 5 55   
## [1] "V6"  
##   
## 1 2 3 4 5 6 7 8 9 10   
## 47 386 72 48 39 41 12 21 2 31   
## [1] "V7"  
##   
## ? 1 10 2 3 4 5 6 7 8 9   
## 16 402 132 30 28 19 30 4 8 21 9   
## [1] "V8"  
##   
## 1 2 3 4 5 6 7 8 9 10   
## 152 166 165 40 34 10 73 28 11 20   
## [1] "V9"  
##   
## 1 2 3 4 5 6 7 8 9 10   
## 443 36 44 18 19 22 16 24 16 61   
## [1] "V10"  
##   
## 1 2 3 4 5 6 7 8 10   
## 579 35 33 12 6 3 9 8 14   
## [1] "V11"  
##   
## 2 4   
## 458 241

# show column with missing value  
table(bc.data$V7)

##   
## ? 1 10 2 3 4 5 6 7 8 9   
## 16 402 132 30 28 19 30 4 8 21 9

# dataset to be imputed  
bc.data\_impute <- which(bc.data$V7 == "?")  
bc.data\_impute

## [1] 24 41 140 146 159 165 236 250 276 293 295 298 316 322 412 618

# QUantifying the missing data  
percent(length(bc.data\_impute)/nrow(bc.data))

## [1] "2.29%"

# Since missing data is less than 5%, we can proceed with imputing.  
  
# Split data into clean and missing  
  
bc.data.clean <- bc.data[-bc.data\_impute,]  
bc.data.missing <- bc.data[bc.data\_impute,]  
  
# Checking for impact of missing data by comparing whole dataset propotion with Clean data set propotion and misisng data set propotion  
prop.table(table(bc.data$V11))

##   
## 2 4   
## 0.6552217 0.3447783

prop.table(table(bc.data.clean$V11))

##   
## 2 4   
## 0.6500732 0.3499268

prop.table(table(bc.data.missing$V11))

##   
## 2 4   
## 0.875 0.125

# 14.1.1 Imputing missing values with Mode/Mean

# Although we are using numeric values but the model is classifying in the end, so I will compute both Mode and Median.  
  
# Calculating Mode  
  
# Function for calculating mode  
  
Mode <- function(x) {  
 ux <- unique(x)  
 ux[which.max(tabulate(match(x, ux)))]  
}  
  
V7.mode <- Mode(bc.data.clean[,'V7'])  
  
print(paste0("Mode for V7 Column: ", V7.mode))

## [1] "Mode for V7 Column: 1"

# Imputing the mode values   
  
bc.data.impute.mode <- bc.data  
bc.data.impute.mode[bc.data\_impute,]$V7<- V7.mode  
  
# Check for the replaced values  
table(bc.data.impute.mode$V7)

##   
## 1 10 2 3 4 5 6 7 8 9   
## 418 132 30 28 19 30 4 8 21 9

# All 16 missing values are replaced with the Mode value of 1  
  
# Calculating mean  
  
V7.mean <- mean(as.integer(bc.data.clean[,'V7']))  
  
# Round the value as we shall have integers between 1~10  
round(V7.mean)

## [1] 4

# Imputing the mean values  
  
bc.data.impute.mean <- bc.data  
bc.data.impute.mean[bc.data\_impute,]$V7<- round(V7.mean)  
  
# Check for the replaced values  
table(bc.data.impute.mean$V7)

##   
## 1 10 2 3 4 5 6 7 8 9   
## 402 132 30 28 35 30 4 8 21 9

# 14.2.2 Using Regression model to impute missing data

# Creating linear model excluding the V11 column, model will be based on clean data   
  
bc.data.clean.reg\_model <- lm(V7~.,data = bc.data.clean[,2:10])  
summary(bc.data.clean.reg\_model)

##   
## Call:  
## lm(formula = V7 ~ ., data = bc.data.clean[, 2:10])  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -9.7316 -0.9426 -0.3002 0.6725 8.6998   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.616652 0.194975 -3.163 0.00163 \*\*   
## V2 0.230156 0.041691 5.521 4.83e-08 \*\*\*  
## V3 -0.067980 0.076170 -0.892 0.37246   
## V4 0.340442 0.073420 4.637 4.25e-06 \*\*\*  
## V5 0.339705 0.045919 7.398 4.13e-13 \*\*\*  
## V6 0.090392 0.062541 1.445 0.14883   
## V8 0.320577 0.059047 5.429 7.91e-08 \*\*\*  
## V9 0.007293 0.044486 0.164 0.86983   
## V10 -0.075230 0.059331 -1.268 0.20524   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.274 on 674 degrees of freedom  
## Multiple R-squared: 0.615, Adjusted R-squared: 0.6104   
## F-statistic: 134.6 on 8 and 674 DF, p-value: < 2.2e-16

# Calculate the AIC to select the most relevant attributes and refine the model  
step(bc.data.clean.reg\_model)

## Start: AIC=1131.43  
## V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10  
##   
## Df Sum of Sq RSS AIC  
## - V9 1 0.139 3486.8 1129.5  
## - V3 1 4.120 3490.8 1130.2  
## - V10 1 8.317 3495.0 1131.0  
## <none> 3486.6 1131.4  
## - V6 1 10.806 3497.5 1131.5  
## - V4 1 111.227 3597.9 1150.9  
## - V8 1 152.482 3639.1 1158.7  
## - V2 1 157.657 3644.3 1159.6  
## - V5 1 283.119 3769.8 1182.8  
##   
## Step: AIC=1129.45  
## V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V10  
##   
## Df Sum of Sq RSS AIC  
## - V3 1 4.028 3490.8 1128.2  
## - V10 1 8.179 3495.0 1129.0  
## <none> 3486.8 1129.5  
## - V6 1 11.211 3498.0 1129.7  
## - V4 1 114.768 3601.6 1149.6  
## - V2 1 158.696 3645.5 1157.8  
## - V8 1 160.776 3647.6 1158.2  
## - V5 1 285.902 3772.7 1181.3  
##   
## Step: AIC=1128.24  
## V7 ~ V2 + V4 + V5 + V6 + V8 + V10  
##   
## Df Sum of Sq RSS AIC  
## - V6 1 8.606 3499.4 1127.9  
## - V10 1 8.889 3499.7 1128.0  
## <none> 3490.8 1128.2  
## - V4 1 153.078 3643.9 1155.6  
## - V2 1 155.308 3646.1 1156.0  
## - V8 1 157.123 3647.9 1156.3  
## - V5 1 282.133 3772.9 1179.3  
##   
## Step: AIC=1127.92  
## V7 ~ V2 + V4 + V5 + V8 + V10  
##   
## Df Sum of Sq RSS AIC  
## - V10 1 5.562 3505.0 1127.0  
## <none> 3499.4 1127.9  
## - V2 1 159.594 3659.0 1156.4  
## - V8 1 169.954 3669.4 1158.3  
## - V4 1 206.785 3706.2 1165.1  
## - V5 1 295.807 3795.2 1181.3  
##   
## Step: AIC=1127.01  
## V7 ~ V2 + V4 + V5 + V8  
##   
## Df Sum of Sq RSS AIC  
## <none> 3505.0 1127.0  
## - V2 1 155.70 3660.7 1154.7  
## - V8 1 172.42 3677.4 1157.8  
## - V4 1 201.22 3706.2 1163.1  
## - V5 1 290.68 3795.7 1179.4

##   
## Call:  
## lm(formula = V7 ~ V2 + V4 + V5 + V8, data = bc.data.clean[, 2:10])  
##   
## Coefficients:  
## (Intercept) V2 V4 V5 V8   
## -0.5360 0.2262 0.3173 0.3323 0.3238

# Refining the model based on AIC values  
  
bc.data.clean.reg\_model\_refined <- lm(V7~ V2 + V4 + V5 + V8, data = bc.data.clean[,2:10])  
summary(bc.data.clean.reg\_model\_refined)

##   
## Call:  
## lm(formula = V7 ~ V2 + V4 + V5 + V8, data = bc.data.clean[, 2:10])  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -9.8115 -0.9531 -0.3111 0.6678 8.6889   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.53601 0.17514 -3.060 0.0023 \*\*   
## V2 0.22617 0.04121 5.488 5.75e-08 \*\*\*  
## V4 0.31729 0.05086 6.239 7.76e-10 \*\*\*  
## V5 0.33227 0.04431 7.499 2.03e-13 \*\*\*  
## V8 0.32378 0.05606 5.775 1.17e-08 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.274 on 678 degrees of freedom  
## Multiple R-squared: 0.6129, Adjusted R-squared: 0.6107   
## F-statistic: 268.4 on 4 and 678 DF, p-value: < 2.2e-16

# Running prediction for missing values  
  
V7.lm\_refined <- predict(bc.data.clean.reg\_model\_refined, bc.data.missing)  
V7.lm\_refined

## 24 41 140 146 159 165 236   
## 5.4585352 7.9816106 0.9872832 1.6218560 0.9807851 2.2157441 2.7152652   
## 250 276 293 295 298 316 322   
## 1.7634059 2.0741942 6.0866099 0.9872832 2.5265324 5.2438347 1.7634059   
## 412 618   
## 0.9872832 0.6634986

round(V7.lm\_refined)

## 24 41 140 146 159 165 236 250 276 293 295 298 316 322 412 618   
## 5 8 1 2 1 2 3 2 2 6 1 3 5 2 1 1

# Imputing the model based values  
  
bc.data.impute.lm\_refined <- bc.data  
bc.data.impute.lm\_refined[bc.data\_impute,]$V7<- round(V7.lm\_refined)  
  
# Check for the replaced values  
table(bc.data.impute.lm\_refined$V7)

##   
## 1 10 2 3 4 5 6 7 8 9   
## 407 132 35 30 19 32 5 8 22 9

# 14.1.3: Using Regression with Perturbation to impute values

# Generating the random values based on regression model and standard deviation  
V7.lm\_refined.perturb <- rnorm(nrow(bc.data.missing), V7.lm\_refined, sd(V7.lm\_refined))  
  
V7.lm\_refined.perturb

## [1] 4.5843999 4.2710726 0.2100933 4.1919400 -0.5471158 -0.6679199  
## [7] 3.1154711 0.9940353 1.8469035 6.1078161 2.6239186 1.2170485  
## [13] 5.6726957 -1.0177133 1.5999509 -2.3691021

# Observed negative values, each of those to be set to 1 i.e. the minimum possible value in acceptable range 1~9.  
  
# Imputing the perturbed values, and then converting negative values to 1.  
  
bc.data.impute.lm\_refined\_perturb <- bc.data  
bc.data.impute.lm\_refined\_perturb[bc.data\_impute,]$V7<- round(V7.lm\_refined.perturb)  
  
table(bc.data.impute.lm\_refined\_perturb$V7)

##   
## -1 -2 0 1 10 2 3 4 5 6 7 8 9   
## 3 1 1 404 132 32 30 21 31 6 8 21 9

bc.data.impute.lm\_refined\_perturb$V7[bc.data.impute.lm\_refined\_perturb$V7 < 1] <- 1  
  
# Confirm if the changes have been implemented  
table(bc.data.impute.lm\_refined\_perturb$V7)

##   
## 1 10 2 3 4 5 6 7 8 9   
## 409 132 32 30 21 31 6 8 21 9

# 14.1.4.1: Comparing results and quality of classification from question 1,2,3

# Spliting the data  
  
training <- sample(nrow(bc.data), size = floor(nrow(bc.data) \* 0.7))  
validation <- setdiff(1:nrow(bc.data), training)  
  
# For mode imputation  
  
for (k in 1:5) {  
   
 knn\_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10, bc.data.impute.mode[training,], bc.data.impute.mode[validation,], k=k)  
   
 pred <- as.integer(fitted(knn\_model)+0.5) # round off to 2 or 4  
   
 acc\_knn = sum(pred == bc.data.impute.mode[validation,]$V11) / nrow(bc.data.impute.mode[validation,])  
 print(acc\_knn)  
}

## [1] 0.9571429  
## [1] 0.9571429  
## [1] 0.9380952  
## [1] 0.9380952  
## [1] 0.9333333

# For Mean Impute  
  
for (k in 1:5) {  
   
 knn\_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10, bc.data.impute.mean[training,], bc.data.impute.mean[validation,], k=k)  
   
 pred <- as.integer(fitted(knn\_model)+0.5) # round off to 2 or 4  
   
 acc\_knn = sum(pred == bc.data.impute.mean[validation,]$V11) / nrow(bc.data.impute.mean[validation,])  
 print(acc\_knn)  
}

## [1] 0.9571429  
## [1] 0.9571429  
## [1] 0.9380952  
## [1] 0.9380952  
## [1] 0.9285714

# For Regression Impute  
  
for (k in 1:5) {  
   
 knn\_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10, bc.data.impute.lm\_refined[training,], bc.data.impute.lm\_refined[validation,], k=k)  
   
 pred <- as.integer(fitted(knn\_model)+0.5) # round off to 2 or 4  
   
 acc\_knn = sum(pred == bc.data.impute.lm\_refined[validation,]$V11) / nrow(bc.data.impute.lm\_refined[validation,])  
 print(acc\_knn)  
}

## [1] 0.952381  
## [1] 0.952381  
## [1] 0.9333333  
## [1] 0.9333333  
## [1] 0.9285714

# For Perturbed Regression Impute  
  
for (k in 1:5) {  
   
 knn\_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10, bc.data.impute.lm\_refined\_perturb[training,], bc.data.impute.lm\_refined\_perturb[validation,], k=k)  
   
 pred <- as.integer(fitted(knn\_model)+0.5) # round off to 2 or 4  
   
 acc\_knn = sum(pred == bc.data.impute.lm\_refined\_perturb[validation,]$V11) / nrow(bc.data.impute.lm\_refined\_perturb[validation,])  
 print(acc\_knn)  
}

## [1] 0.9571429  
## [1] 0.9571429  
## [1] 0.9333333  
## [1] 0.9333333  
## [1] 0.9285714

# 14.1.4.2: Comparing results and quality of classification with missing values removed

# Creating a new split as the cleaned data has fewer number of rows hence the reference number of rows will change  
training\_clean <- sample(nrow(bc.data.clean), size = floor(nrow(bc.data.clean) \* 0.7))  
validation\_clean <- setdiff(1:nrow(bc.data.clean), training)  
  
# For clean data (excluding missing(?) data rows)  
for (k in 1:5) {  
   
 knn\_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10, bc.data.clean[training\_clean,], bc.data.clean[validation\_clean,], k=k)  
   
 pred <- as.integer(fitted(knn\_model)+0.5) # round off to 2 or 4  
   
 acc\_knn = sum(pred == bc.data.clean[validation\_clean,]$V11) / nrow(bc.data.clean[validation\_clean,])  
 print(acc\_knn)  
}

## [1] 0.9852941  
## [1] 0.9852941  
## [1] 0.9656863  
## [1] 0.9656863  
## [1] 0.9656863

# 14.1.4.2: Comparing results and quality of classification with binary variables

# Creating a new dataset for binary interaction  
bc.data.binary<- bc.data  
  
# mutating Columns for binary interaction between V7 and V12 under V13  
bc.data.binary <- bc.data %>%  
 mutate(V12 = if\_else(bc.data$V7 == "?",0,1))%>%  
 mutate(V13 = ifelse(V12 == 0, 0, paste0(bc.data.binary$V7)))  
  
# For data with binary interaction  
for (k in 1:5) {  
   
 knn\_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10, bc.data.binary[training,], bc.data.binary[validation,], k=k)  
   
 pred <- as.integer(fitted(knn\_model)+0.5) # round off to 2 or 4  
   
 acc\_knn = sum(pred == bc.data.binary[validation,]$V11) / nrow(bc.data.binary[validation,])  
 print(acc\_knn)  
}

## [1] 0.9571429  
## [1] 0.9571429  
## [1] 0.9380952  
## [1] 0.9380952  
## [1] 0.9285714

Conclusions

* **Model accuracy improves with clean data, apart from that all other methods provide the accuracy in very close range (92.8-95.7%)**
* **Mode approach is not very accurate** as it assigns all missing values to be 1, whereas with every other method the values were varying above 1
* With **pertrubation some negative values were introduced**, those had to set to 1 (minimum in the acceptable range of 1~9)