PROG8430-Assignment 4

Mohammed Saifullah

2023-04-13

Loading necessary packages

#Load packages  
if(!require(tinytex)){install.packages("tinytex")}

## Loading required package: tinytex

library("tinytex")  
  
if(!require(pastecs)){install.packages("pastecs")}

## Loading required package: pastecs

library("pastecs")  
  
if(!require(lattice)){install.packages("lattice")}

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 4.2.3

library("lattice")  
  
if(!require(vcd)){install.packages("vcd")}

## Loading required package: vcd

## Warning: package 'vcd' was built under R version 4.2.3

## Loading required package: grid

library("vcd")  
  
if(!require(HSAUR)){install.packages("HSAUR")}

## Loading required package: HSAUR

## Loading required package: tools

library("HSAUR")  
  
if(!require(rmarkdown)){install.packages("rmarkdown")}

## Loading required package: rmarkdown

## Warning: package 'rmarkdown' was built under R version 4.2.3

library("rmarkdown")  
  
if(!require(ggplot2)){install.packages("ggplot2")}

## Loading required package: ggplot2

library("ggplot2")  
  
if(!require(polycor)){install.packages("polycor")}

## Loading required package: polycor

## Warning: package 'polycor' was built under R version 4.2.3

library("polycor")  
  
  
if(!require(klaR)){install.packages("klaR")}

## Loading required package: klaR

## Warning: package 'klaR' was built under R version 4.2.3

## Loading required package: MASS

library("klaR")  
  
if(!require(MASS)){install.packages("MASS")}  
library("MASS")  
  
if(!require(partykit)){install.packages("partykit")}

## Loading required package: partykit

## Warning: package 'partykit' was built under R version 4.2.3

## Loading required package: libcoin

## Warning: package 'libcoin' was built under R version 4.2.3

## Loading required package: mvtnorm

library("partykit")  
  
if(!require(nnet)){install.packages("nnet")}

## Loading required package: nnet

library("nnet")  
  
if(!require(corrgram)){install.packages("corrgram")}

## Loading required package: corrgram

##   
## Attaching package: 'corrgram'

## The following object is masked from 'package:lattice':  
##   
## panel.fill

library("corrgram")

# **Part A - Preliminary Data Preparation**

##1.1 Appending initials to all column names

getwd() #verify working directory

## [1] "E:/Big Data Solution Architecture/PROG8430 - Data Analysis Mathematics, Algorithms and Modeling/Assignment 4"

#Read the text data file into a Data Frame  
MailOrder\_MS <- read.table("PROG8430\_Assign04\_23W.txt", sep=',', header = TRUE)  
#concatenating initial 'MS' to all column names  
colnames(MailOrder\_MS) <- paste(colnames(MailOrder\_MS), "MS", sep = "\_")  
#Display first 5 rows of the dataset just to verify loading and name transformation is successful  
head(MailOrder\_MS, 5)

## DL\_MS VN\_MS PG\_MS CS\_MS ML\_MS DM\_MS HZ\_MS CR\_MS WT\_MS  
## 1 8.1 324 5 13 313 C N Sup Del 216  
## 2 8.4 135 2 13 830 I N Sup Del 160  
## 3 8.6 391 3 12 304 C N Sup Del 25  
## 4 11.3 245 6 7 1258 C N Sup Del 67  
## 5 5.4 321 1 2 221 C N Def Post 14

#Transform String as Factor variable  
MailOrder\_MS <- as.data.frame(unclass(MailOrder\_MS), stringsAsFactors = TRUE)  
#Checking Data Structure  
str(MailOrder\_MS)

## 'data.frame': 487 obs. of 9 variables:  
## $ DL\_MS: num 8.1 8.4 8.6 11.3 5.4 9.4 8.2 9.4 9.3 9.7 ...  
## $ VN\_MS: int 324 135 391 245 321 397 390 252 355 159 ...  
## $ PG\_MS: int 5 2 3 6 1 2 6 2 4 1 ...  
## $ CS\_MS: int 13 13 12 7 2 8 13 8 2 12 ...  
## $ ML\_MS: int 313 830 304 1258 221 1002 655 1367 675 888 ...  
## $ DM\_MS: Factor w/ 2 levels "C","I": 1 2 1 1 1 2 1 2 1 1 ...  
## $ HZ\_MS: Factor w/ 2 levels "H","N": 2 2 2 2 2 2 2 2 2 2 ...  
## $ CR\_MS: Factor w/ 2 levels "Def Post","Sup Del": 2 2 2 2 1 2 2 2 2 2 ...  
## $ WT\_MS: num 216 160 25 67 14 47 7 6 30 177 ...

##1.2 Deleting the observation with PG\_MS < 0

MailOrder\_MS <- MailOrder\_MS[!MailOrder\_MS$PG\_MS < 0,]  
  
summary(MailOrder\_MS)

## DL\_MS VN\_MS PG\_MS CS\_MS   
## Min. : 1.800 Min. : 85.0 Min. :1.000 Min. : 0.000   
## 1st Qu.: 7.400 1st Qu.:263.0 1st Qu.:2.000 1st Qu.: 5.000   
## Median : 8.500 Median :322.0 Median :3.000 Median : 8.000   
## Mean : 8.464 Mean :318.7 Mean :2.961 Mean : 9.228   
## 3rd Qu.: 9.575 3rd Qu.:371.0 3rd Qu.:4.000 3rd Qu.:13.000   
## Max. :14.400 Max. :495.0 Max. :9.000 Max. :24.000   
## ML\_MS DM\_MS HZ\_MS CR\_MS WT\_MS   
## Min. : 35.0 C:343 H: 68 Def Post:201 Min. : 0.1   
## 1st Qu.: 444.2 I:143 N:418 Sup Del :285 1st Qu.: 33.0   
## Median : 697.5 Median : 86.5   
## Mean : 754.2 Mean :107.1   
## 3rd Qu.:1021.8 3rd Qu.:157.8   
## Max. :1967.0 Max. :500.0

##1.3 Creating a new variable in the dataset called OT\_MS which will have a value of 1 if DL\_MS ≤ 8.5 and 0 otherwise

MailOrder\_MS$OT\_MS <- as.factor(ifelse(MailOrder\_MS$DL\_MS <= 8.5, 1,0))  
  
#Delete the DL\_MS variable  
MailOrder\_MS <- MailOrder\_MS[,-c(1)]  
head(MailOrder\_MS)

## VN\_MS PG\_MS CS\_MS ML\_MS DM\_MS HZ\_MS CR\_MS WT\_MS OT\_MS  
## 1 324 5 13 313 C N Sup Del 216 1  
## 2 135 2 13 830 I N Sup Del 160 1  
## 3 391 3 12 304 C N Sup Del 25 0  
## 4 245 6 7 1258 C N Sup Del 67 0  
## 5 321 1 2 221 C N Def Post 14 1  
## 6 397 2 8 1002 I N Sup Del 47 0

## Split the data set into Training and Test set

#Choosing sampling rate for training data  
sr\_ms <- 0.8 #80% in training set  
  
# Finding the number of rows of data  
n.row <- nrow(MailOrder\_MS) #counting number of rows  
  
#Choose the rows for the training sample   
  
set.seed(6024) #setting a seed, same starting point. Last 4 digits of my student ID  
training.rows <- sample(1:n.row, sr\_ms\*n.row, replace=FALSE) #sampling   
#selecting from 1 to no of rows, how much - sampling-rate\*no or rows, placement equal false - don't want to replace  
  
#Assigning to the training sample  
train\_ms <- subset(MailOrder\_MS[training.rows,]) #creating training data set, only keeping training rows  
  
# Assign the balance to the Test Sample  
  
test\_ms <- subset(MailOrder\_MS[-c(training.rows),]) #keeping everything except training rows  
  
#Checking Train and Test datasets  
head(training.rows)

## [1] 195 305 179 116 157 316

head(train\_ms)

## VN\_MS PG\_MS CS\_MS ML\_MS DM\_MS HZ\_MS CR\_MS WT\_MS OT\_MS  
## 196 346 1 1 938 I N Def Post 118 1  
## 306 256 2 8 1009 I N Sup Del 2 0  
## 180 371 2 7 697 C N Sup Del 56 1  
## 116 461 3 4 1243 C N Sup Del 90 0  
## 157 368 3 10 633 I N Def Post 8 0  
## 317 345 5 12 196 C N Sup Del 81 0

head(test\_ms)

## VN\_MS PG\_MS CS\_MS ML\_MS DM\_MS HZ\_MS CR\_MS WT\_MS OT\_MS  
## 1 324 5 13 313 C N Sup Del 216 1  
## 2 135 2 13 830 I N Sup Del 160 1  
## 5 321 1 2 221 C N Def Post 14 1  
## 26 354 1 5 181 I N Sup Del 8 1  
## 29 357 5 10 684 C N Sup Del 130 0  
## 30 354 1 2 576 C H Sup Del 95 1

summary(MailOrder\_MS)

## VN\_MS PG\_MS CS\_MS ML\_MS DM\_MS   
## Min. : 85.0 Min. :1.000 Min. : 0.000 Min. : 35.0 C:343   
## 1st Qu.:263.0 1st Qu.:2.000 1st Qu.: 5.000 1st Qu.: 444.2 I:143   
## Median :322.0 Median :3.000 Median : 8.000 Median : 697.5   
## Mean :318.7 Mean :2.961 Mean : 9.228 Mean : 754.2   
## 3rd Qu.:371.0 3rd Qu.:4.000 3rd Qu.:13.000 3rd Qu.:1021.8   
## Max. :495.0 Max. :9.000 Max. :24.000 Max. :1967.0   
## HZ\_MS CR\_MS WT\_MS OT\_MS   
## H: 68 Def Post:201 Min. : 0.1 0:233   
## N:418 Sup Del :285 1st Qu.: 33.0 1:253   
## Median : 86.5   
## Mean :107.1   
## 3rd Qu.:157.8   
## Max. :500.0

summary(test\_ms)

## VN\_MS PG\_MS CS\_MS ML\_MS DM\_MS   
## Min. :135.0 Min. :1.000 Min. : 1.000 Min. : 97.0 C:69   
## 1st Qu.:271.2 1st Qu.:2.000 1st Qu.: 5.000 1st Qu.: 445.5 I:29   
## Median :318.0 Median :3.000 Median : 9.000 Median : 717.0   
## Mean :312.1 Mean :3.184 Mean : 9.633 Mean : 755.9   
## 3rd Qu.:364.0 3rd Qu.:4.750 3rd Qu.:14.000 3rd Qu.:1033.0   
## Max. :483.0 Max. :8.000 Max. :24.000 Max. :1807.0   
## HZ\_MS CR\_MS WT\_MS OT\_MS   
## H:11 Def Post:30 Min. : 2.0 0:44   
## N:87 Sup Del :68 1st Qu.: 39.5 1:54   
## Median : 99.5   
## Mean :116.8   
## 3rd Qu.:168.8   
## Max. :452.0

summary(train\_ms)

## VN\_MS PG\_MS CS\_MS ML\_MS DM\_MS   
## Min. : 85.0 Min. :1.000 Min. : 0.000 Min. : 35.0 C:274   
## 1st Qu.:261.5 1st Qu.:2.000 1st Qu.: 5.000 1st Qu.: 442.0 I:114   
## Median :324.0 Median :3.000 Median : 8.000 Median : 695.5   
## Mean :320.3 Mean :2.905 Mean : 9.126 Mean : 753.7   
## 3rd Qu.:373.2 3rd Qu.:4.000 3rd Qu.:13.000 3rd Qu.:1012.2   
## Max. :495.0 Max. :9.000 Max. :24.000 Max. :1967.0   
## HZ\_MS CR\_MS WT\_MS OT\_MS   
## H: 57 Def Post:171 Min. : 0.10 0:189   
## N:331 Sup Del :217 1st Qu.: 31.75 1:199   
## Median : 83.50   
## Mean :104.67   
## 3rd Qu.:151.25   
## Max. :500.00

# **Part 2 - Exploratory Analysis**

##2.1 Checking Correlations

#Numeric Correlation  
ht\_ms <- hetcor(MailOrder\_MS) #heterogeneous correlation  
  
round(ht\_ms$correlations, 2)

## VN\_MS PG\_MS CS\_MS ML\_MS DM\_MS HZ\_MS CR\_MS WT\_MS OT\_MS  
## VN\_MS 1.00 0.01 -0.02 -0.01 0.05 0.04 -0.07 0.00 0.00  
## PG\_MS 0.01 1.00 0.08 0.06 0.03 0.08 -0.04 -0.01 -0.50  
## CS\_MS -0.02 0.08 1.00 -0.03 0.10 -0.03 0.02 -0.02 -0.04  
## ML\_MS -0.01 0.06 -0.03 1.00 -0.06 -0.05 0.09 -0.04 -0.20  
## DM\_MS 0.05 0.03 0.10 -0.06 1.00 0.05 0.03 0.00 -0.11  
## HZ\_MS 0.04 0.08 -0.03 -0.05 0.05 1.00 0.14 0.11 0.19  
## CR\_MS -0.07 -0.04 0.02 0.09 0.03 0.14 1.00 -0.10 -0.39  
## WT\_MS 0.00 -0.01 -0.02 -0.04 0.00 0.11 -0.10 1.00 0.37  
## OT\_MS 0.00 -0.50 -0.04 -0.20 -0.11 0.19 -0.39 0.37 1.00

**Interpretation:** There are no significant correlation except PG\_MS and OT\_MS, which is inversely correlated.

# **Part 3 - Model Development**

##3.1 Full Model

full.model\_ms <- glm(OT\_MS ~ . , data = train\_ms, family="binomial", na.action = na.omit)  
  
summary(full.model\_ms)

##   
## Call:  
## glm(formula = OT\_MS ~ ., family = "binomial", data = train\_ms,   
## na.action = na.omit)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.2177 -0.8215 0.1996 0.8091 2.6587   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 1.7466015 0.7879306 2.217 0.026644 \*   
## VN\_MS 0.0008483 0.0017491 0.485 0.627708   
## PG\_MS -0.8440357 0.1094104 -7.714 1.22e-14 \*\*\*  
## CS\_MS 0.0089696 0.0253667 0.354 0.723641   
## ML\_MS -0.0007694 0.0003157 -2.437 0.014799 \*   
## DM\_MSI -0.4264837 0.2834246 -1.505 0.132388   
## HZ\_MSN 1.2142088 0.3676624 3.303 0.000958 \*\*\*  
## CR\_MSSup Del -1.4039282 0.2698236 -5.203 1.96e-07 \*\*\*  
## WT\_MS 0.0084385 0.0016184 5.214 1.85e-07 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 537.62 on 387 degrees of freedom  
## Residual deviance: 381.71 on 379 degrees of freedom  
## AIC: 399.71  
##   
## Number of Fisher Scoring iterations: 5

**Interpretation:** Comparing following measures for full model.  
(1) Fisher iterations - 4, the model converged in reasonable iterations.  
(2) AIC - 399.71  
(3) Residual Deviance - Null daviance of 537.62 and Residual deviance of 381.71 are not close, so this model is capable of prediction.  
(4) Residual symmetry - Residuals are centered around 0 and symmetric.  
(5) z-values - 5/8 variable have p value less than 0.05, so 5 variable out of 8 passed z-test.  
(6) Parameter Co-Efficients - 7 out of 8 variable have coefficient matching to the correlation matrix.

##3.2 Back Model

back.model\_ms = step(full.model\_ms, direction="backward", details=TRUE)

## Start: AIC=399.71  
## OT\_MS ~ VN\_MS + PG\_MS + CS\_MS + ML\_MS + DM\_MS + HZ\_MS + CR\_MS +   
## WT\_MS  
##   
## Df Deviance AIC  
## - CS\_MS 1 381.84 397.84  
## - VN\_MS 1 381.95 397.95  
## <none> 381.71 399.71  
## - DM\_MS 1 384.00 400.00  
## - ML\_MS 1 387.78 403.78  
## - HZ\_MS 1 393.19 409.19  
## - CR\_MS 1 411.28 427.28  
## - WT\_MS 1 414.81 430.81  
## - PG\_MS 1 464.25 480.25  
##   
## Step: AIC=397.84  
## OT\_MS ~ VN\_MS + PG\_MS + ML\_MS + DM\_MS + HZ\_MS + CR\_MS + WT\_MS  
##   
## Df Deviance AIC  
## - VN\_MS 1 382.07 396.07  
## <none> 381.84 397.84  
## - DM\_MS 1 384.08 398.08  
## - ML\_MS 1 387.98 401.98  
## - HZ\_MS 1 393.23 407.23  
## - CR\_MS 1 411.48 425.48  
## - WT\_MS 1 414.82 428.82  
## - PG\_MS 1 464.53 478.53  
##   
## Step: AIC=396.07  
## OT\_MS ~ PG\_MS + ML\_MS + DM\_MS + HZ\_MS + CR\_MS + WT\_MS  
##   
## Df Deviance AIC  
## <none> 382.07 396.07  
## - DM\_MS 1 384.20 396.20  
## - ML\_MS 1 388.21 400.21  
## - HZ\_MS 1 393.59 405.59  
## - CR\_MS 1 411.86 423.86  
## - WT\_MS 1 414.84 426.84  
## - PG\_MS 1 464.53 476.53

summary(back.model\_ms)

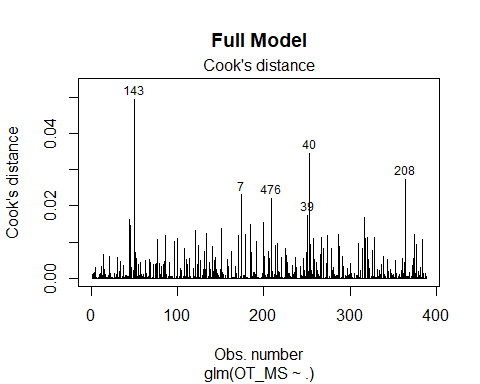
##   
## Call:  
## glm(formula = OT\_MS ~ PG\_MS + ML\_MS + DM\_MS + HZ\_MS + CR\_MS +   
## WT\_MS, family = "binomial", data = train\_ms, na.action = na.omit)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.2505 -0.8146 0.2091 0.8037 2.6851   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 2.0919280 0.5277509 3.964 7.37e-05 \*\*\*  
## PG\_MS -0.8376294 0.1086089 -7.712 1.24e-14 \*\*\*  
## ML\_MS -0.0007725 0.0003151 -2.452 0.014222 \*   
## DM\_MSI -0.4085979 0.2812247 -1.453 0.146245   
## HZ\_MSN 1.2106167 0.3659184 3.308 0.000938 \*\*\*  
## CR\_MSSup Del -1.4065781 0.2693996 -5.221 1.78e-07 \*\*\*  
## WT\_MS 0.0083439 0.0016020 5.208 1.91e-07 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 537.62 on 387 degrees of freedom  
## Residual deviance: 382.07 on 381 degrees of freedom  
## AIC: 396.07  
##   
## Number of Fisher Scoring iterations: 5

**Interpretation:** Comparing following measures for backward selection model.

Comparing following measures for full model.  
(1) Fisher iterations - 5, the model converged in reasonable iterations.  
(2) AIC - 396.07  
(3) Residual Deviance - Null daviance of 537.62 and Residual deviance of 382.07 are not close, so this model is capable of prediction.  
(4) Residual symmetry - Residuals are centered around 0 and symmetric.  
(5) z-values - 5/6 variable have p value less than 0.05, so 5 variable out of 6 passed z-test.  
(6) Parameter Co-Efficients - 6 out of 6 variable have coefficient matching to the correlation matrix.

##3.3 Influential Datapoint  
#Full Model Analysis

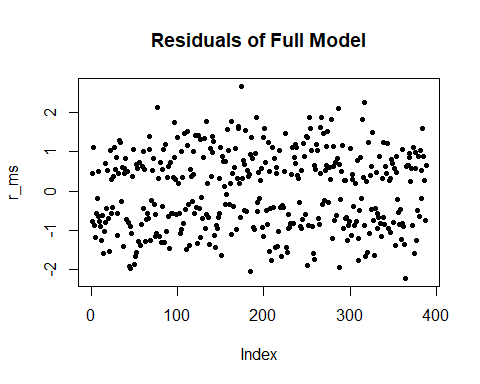
plot(full.model\_ms, which=4, id.n=6, main = "Full Model")



r\_ms <- residuals(full.model\_ms)  
head(r\_ms)

## 196 306 180 116 157 317   
## 0.4441172 -0.7819731 1.1009353 -0.8813589 -1.1708449 -0.5712526

plot(r\_ms, pch=20, main = "Residuals of Full Model")



#Confusion Matrix and measures  
resp\_glm\_ms <- predict(full.model\_ms, newdata=train\_ms, type="response")   
Class\_glm\_ms <- ifelse(resp\_glm\_ms > 0.5,"1","0")   
CF\_GLM\_MS <- table(train\_ms$OT\_MS, Class\_glm\_ms,  
 dnn=list("Actual","Predicted") )   
  
CF\_GLM\_MS

## Predicted  
## Actual 0 1  
## 0 141 48  
## 1 46 153

TP <- CF\_GLM\_MS[2,2]  
TN <- CF\_GLM\_MS[1,1]  
FP <- CF\_GLM\_MS[1,2]  
FN <- CF\_GLM\_MS[2,1]   
  
#Calculate Accuracy  
Tr\_Accuracy\_ms <- ((TP+TN)/sum(CF\_GLM\_MS))  
sprintf("Accuracy: %.3f", round(Tr\_Accuracy\_ms, 3))

## [1] "Accuracy: 0.758"

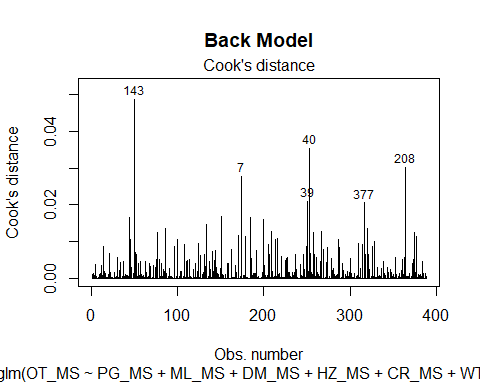
#Calculate Precision  
Tr\_Precision\_ms <- TP/(TP+FP)  
sprintf("Precision: %.3f", round(Tr\_Precision\_ms, 3))

## [1] "Precision: 0.761"

**Interpretation:** From the plot we see that all data points are within cook’s distance, so there are no significant influential data points.

#Back Model Analysis

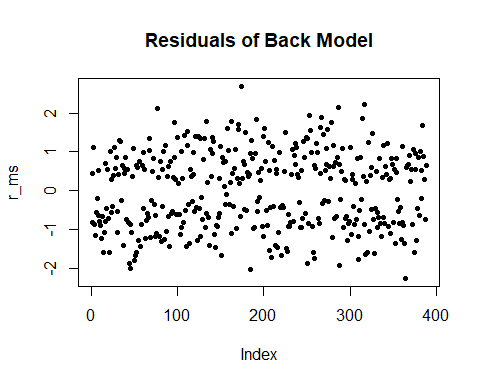
plot(back.model\_ms, which=4, id.n=6, main = "Back Model")



r\_ms <- residuals(back.model\_ms)  
head(r\_ms)

## 196 306 180 116 157 317   
## 0.4337531 -0.8084285 1.1146019 -0.8521264 -1.1599764 -0.5614593

plot(r\_ms, pch=20, main = "Residuals of Back Model")



#Confusion Matrix and measures  
resp\_glm\_ms <- predict(back.model\_ms, newdata=train\_ms, type="response")   
Class\_glm\_ms <- ifelse(resp\_glm\_ms > 0.5,"1","0")   
CF\_GLM\_MS <- table(train\_ms$OT\_MS, Class\_glm\_ms,  
 dnn=list("Actual","Predicted") )   
  
CF\_GLM\_MS

## Predicted  
## Actual 0 1  
## 0 141 48  
## 1 46 153

TP <- CF\_GLM\_MS[2,2]  
TN <- CF\_GLM\_MS[1,1]  
FP <- CF\_GLM\_MS[1,2]  
FN <- CF\_GLM\_MS[2,1]   
  
#Calculate Accuracy  
Tr\_Accuracy\_ms <- ((TP+TN)/sum(CF\_GLM\_MS))  
sprintf("Accuracy: %.3f", round(Tr\_Accuracy\_ms, 3))

## [1] "Accuracy: 0.758"

#Calculate Precision  
Tr\_Precision\_ms <- TP/(TP+FP)  
sprintf("Precision: %.3f", round(Tr\_Precision\_ms, 3))

## [1] "Precision: 0.761"

**Interpretation:** From the plot we see that all data points are within cook’s distance, so there are no significant influential data points.

##3.4 Recommendation Based on the above measure I recommend backward model as the superior one.Because :  
- Both model have same accuracy and precision  
- Backward model coefficient matching 6/6  
- 5/6 variable have p value less than 0.05 for backward model as oppose to 5/8 for full model.

# **Part B**

## 1 Logistic Regression - Stepwise

start\_time <- Sys.time()  
   
glm.mod\_ms = glm(OT\_MS ~ . ,  
 family="binomial", data=train\_ms, na.action=na.omit)  
   
stp\_model\_ms <- step(glm.mod\_ms, trace=FALSE)  
   
end\_time <- Sys.time()  
   
time\_ms <- end\_time - start\_time #model time  
   
summary(stp\_model\_ms)

##   
## Call:  
## glm(formula = OT\_MS ~ PG\_MS + ML\_MS + DM\_MS + HZ\_MS + CR\_MS +   
## WT\_MS, family = "binomial", data = train\_ms, na.action = na.omit)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.2505 -0.8146 0.2091 0.8037 2.6851   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 2.0919280 0.5277509 3.964 7.37e-05 \*\*\*  
## PG\_MS -0.8376294 0.1086089 -7.712 1.24e-14 \*\*\*  
## ML\_MS -0.0007725 0.0003151 -2.452 0.014222 \*   
## DM\_MSI -0.4085979 0.2812247 -1.453 0.146245   
## HZ\_MSN 1.2106167 0.3659184 3.308 0.000938 \*\*\*  
## CR\_MSSup Del -1.4065781 0.2693996 -5.221 1.78e-07 \*\*\*  
## WT\_MS 0.0083439 0.0016020 5.208 1.91e-07 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 537.62 on 387 degrees of freedom  
## Residual deviance: 382.07 on 381 degrees of freedom  
## AIC: 396.07  
##   
## Number of Fisher Scoring iterations: 5

time\_ms

## Time difference of 0.04823208 secs

# Creating Confusion Matrix on Train Dataset  
   
resp\_glm\_ms <- predict(stp\_model\_ms, newdata=train\_ms, type="response")   
Class\_glm\_ms <- ifelse(resp\_glm\_ms > 0.5,"1","0")   
CF\_GLM\_MS <- table(train\_ms$OT\_MS, Class\_glm\_ms,  
 dnn=list("Actual","Predicted") )   
  
CF\_GLM\_MS

## Predicted  
## Actual 0 1  
## 0 141 48  
## 1 46 153

TP <- CF\_GLM\_MS[2,2]  
TN <- CF\_GLM\_MS[1,1]  
FP <- CF\_GLM\_MS[1,2]  
FN <- CF\_GLM\_MS[2,1]   
  
#Calculate Accuracy  
Tr\_Accuracy\_ms <- ((TP+TN)/sum(CF\_GLM\_MS))  
sprintf("Train Accuracy: %.3f", round(Tr\_Accuracy\_ms, 3))

## [1] "Train Accuracy: 0.758"

#Calculate Precision  
Tr\_Precision\_ms <- TP/(TP+FP)  
sprintf("Train Precision: %.3f", round(Tr\_Precision\_ms, 3))

## [1] "Train Precision: 0.761"

#Calculate Sensitivity  
Tr\_Sensitivity\_ms <- TP/(TP+FN)  
sprintf("Train Sensitivity: %.3f", round(Tr\_Sensitivity\_ms, 3))

## [1] "Train Sensitivity: 0.769"

#Calculate Prevalence  
Tr\_Prevalence\_ms <- (TP+FN)/(sum(CF\_GLM\_MS))  
sprintf("Train Prevalence: %.3f", round(Tr\_Prevalence\_ms, 3))

## [1] "Train Prevalence: 0.513"

#Calculate Specificity  
Tr\_Specificity\_ms <- TN/(TN+FP)  
sprintf("Train Specificity: %.3f", round(Tr\_Specificity\_ms, 3))

## [1] "Train Specificity: 0.746"

# Creating Confusion Matrix on Test Dataset  
   
resp\_glm\_ms <- predict(stp\_model\_ms, newdata=test\_ms, type="response")   
Class\_glm\_ms <- ifelse(resp\_glm\_ms > 0.5,"1","0")   
CF\_GLM\_MS <- table(test\_ms$OT\_MS, Class\_glm\_ms,  
 dnn=list("Actual","Predicted") )   
  
CF\_GLM\_MS

## Predicted  
## Actual 0 1  
## 0 33 11  
## 1 12 42

TP <- CF\_GLM\_MS[2,2]  
TN <- CF\_GLM\_MS[1,1]  
FP <- CF\_GLM\_MS[1,2]  
FN <- CF\_GLM\_MS[2,1]   
  
#Calculate Accuracy  
Tr\_Accuracy\_ms <- ((TP+TN)/sum(CF\_GLM\_MS))  
sprintf("Test Accuracy: %.3f", round(Tr\_Accuracy\_ms, 3))

## [1] "Test Accuracy: 0.765"

#Calculate Precision  
Tr\_Precision\_ms <- TP/(TP+FP)  
sprintf("Test Precision: %.3f", round(Tr\_Precision\_ms, 3))

## [1] "Test Precision: 0.792"

#Calculate Sensitivity  
Tr\_Sensitivity\_ms <- TP/(TP+FN)  
sprintf("Test Sensitivity: %.3f", round(Tr\_Sensitivity\_ms, 3))

## [1] "Test Sensitivity: 0.778"

#Calculate Prevalence  
Tr\_Prevalence\_ms <- (TP+FN)/(sum(CF\_GLM\_MS))  
sprintf("Test Prevalence: %.3f", round(Tr\_Prevalence\_ms, 3))

## [1] "Test Prevalence: 0.551"

#Calculate Specificity  
Tr\_Specificity\_ms <- TN/(TN+FP)  
sprintf("Test Specificity: %.3f", round(Tr\_Specificity\_ms, 3))

## [1] "Test Specificity: 0.750"

**Interpretation:** Between Train and Test dataset Accuracy, Sensitivity and Specificity are very close. precesion and Prevalence is higher in Test data set.

## 2 Naïve-Bayes Classification

start\_time <- Sys.time()  
   
NB.mod\_ms <- NaiveBayes(OT\_MS ~ . ,  
 data = train\_ms, na.action=na.omit)  
   
end\_time <- Sys.time()  
   
time\_ms <- end\_time - start\_time  
   
  
   
time\_ms

## Time difference of 0.003390789 secs

# Creating Confusion Matrix on Train Dataset  
  
pred\_NB\_ms <- predict(NB.mod\_ms,newdata=train\_ms)   
CF\_NB\_MS <- table(Actual=train\_ms$OT\_MS, Predicted=pred\_NB\_ms$class)  
  
  
CF\_NB\_MS

## Predicted  
## Actual 0 1  
## 0 144 45  
## 1 50 149

TP <- CF\_NB\_MS[2,2]  
TN <- CF\_NB\_MS[1,1]  
FP <- CF\_NB\_MS[1,2]  
FN <- CF\_NB\_MS[2,1]   
  
#Calculate Accuracy  
Tr\_Accuracy\_ms <- ((TP+TN)/sum(CF\_NB\_MS))  
sprintf("Train Accuracy: %.3f", round(Tr\_Accuracy\_ms, 3))

## [1] "Train Accuracy: 0.755"

#Calculate Precision  
Tr\_Precision\_ms <- TP/(TP+FP)  
sprintf("Train Precision: %.3f", round(Tr\_Precision\_ms, 3))

## [1] "Train Precision: 0.768"

#Calculate Sensitivity  
Tr\_Sensitivity\_ms <- TP/(TP+FN)  
sprintf("Train Sensitivity: %.3f", round(Tr\_Sensitivity\_ms, 3))

## [1] "Train Sensitivity: 0.749"

#Calculate Prevalence  
Tr\_Prevalence\_ms <- (TP+FN)/(sum(CF\_NB\_MS))  
sprintf("Train Prevalence: %.3f", round(Tr\_Prevalence\_ms, 3))

## [1] "Train Prevalence: 0.513"

#Calculate Specificity  
Tr\_Specificity\_ms <- TN/(TN+FP)  
sprintf("Train Specificity: %.3f", round(Tr\_Specificity\_ms, 3))

## [1] "Train Specificity: 0.762"

# Creating Confusion Matrix on Test Dataset  
   
pred\_NB\_ms <- predict(NB.mod\_ms,newdata=test\_ms)   
CF\_NB\_MS <- table(Actual=test\_ms$OT\_MS, Predicted=pred\_NB\_ms$class)  
  
CF\_NB\_MS

## Predicted  
## Actual 0 1  
## 0 33 11  
## 1 15 39

TP <- CF\_NB\_MS[2,2]  
TN <- CF\_NB\_MS[1,1]  
FP <- CF\_NB\_MS[1,2]  
FN <- CF\_NB\_MS[2,1]   
  
#Calculate Accuracy  
Tr\_Accuracy\_ms <- ((TP+TN)/sum(CF\_NB\_MS))  
sprintf("Test Accuracy: %.3f", round(Tr\_Accuracy\_ms, 3))

## [1] "Test Accuracy: 0.735"

#Calculate Precision  
Tr\_Precision\_ms <- TP/(TP+FP)  
sprintf("Test Precision: %.3f", round(Tr\_Precision\_ms, 3))

## [1] "Test Precision: 0.780"

#Calculate Sensitivity  
Tr\_Sensitivity\_ms <- TP/(TP+FN)  
sprintf("Test Sensitivity: %.3f", round(Tr\_Sensitivity\_ms, 3))

## [1] "Test Sensitivity: 0.722"

#Calculate Prevalence  
Tr\_Prevalence\_ms <- (TP+FN)/(sum(CF\_NB\_MS))  
sprintf("Test Prevalence: %.3f", round(Tr\_Prevalence\_ms, 3))

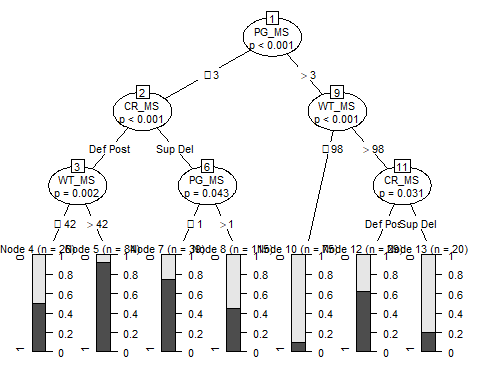
## [1] "Test Prevalence: 0.551"

#Calculate Specificity  
Tr\_Specificity\_ms <- TN/(TN+FP)  
sprintf("Test Specificity: %.3f", round(Tr\_Specificity\_ms, 3))

## [1] "Test Specificity: 0.750"

## 3 Recursive Partitioning Analysis

start\_time <- Sys.time()  
  
RP.mod\_ms <- ctree(OT\_MS ~ ., data=train\_ms)  
  
end\_time <- Sys.time()  
   
time\_ms <- end\_time - start\_time  
  
plot(RP.mod\_ms, gp=gpar(fontsize=8))



time\_ms

## Time difference of 0.03866291 secs

# Creating Confusion Matrix on Train Dataset  
  
pred.RP\_ms <- predict(RP.mod\_ms, newdata=train\_ms)  
CF\_RP\_MS <- table(Actual=train\_ms$OT\_MS, Predicted=pred.RP\_ms)  
  
  
CF\_RP\_MS

## Predicted  
## Actual 0 1  
## 0 161 28  
## 1 75 124

TP <- CF\_RP\_MS[2,2]  
TN <- CF\_RP\_MS[1,1]  
FP <- CF\_RP\_MS[1,2]  
FN <- CF\_RP\_MS[2,1]   
  
#Calculate Accuracy  
Tr\_Accuracy\_ms <- ((TP+TN)/sum(CF\_RP\_MS))  
sprintf("Train Accuracy: %.3f", round(Tr\_Accuracy\_ms, 3))

## [1] "Train Accuracy: 0.735"

#Calculate Precision  
Tr\_Precision\_ms <- TP/(TP+FP)  
sprintf("Train Precision: %.3f", round(Tr\_Precision\_ms, 3))

## [1] "Train Precision: 0.816"

#Calculate Sensitivity  
Tr\_Sensitivity\_ms <- TP/(TP+FN)  
sprintf("Train Sensitivity: %.3f", round(Tr\_Sensitivity\_ms, 3))

## [1] "Train Sensitivity: 0.623"

#Calculate Prevalence  
Tr\_Prevalence\_ms <- (TP+FN)/(sum(CF\_RP\_MS))  
sprintf("Train Prevalence: %.3f", round(Tr\_Prevalence\_ms, 3))

## [1] "Train Prevalence: 0.513"

#Calculate Specificity  
Tr\_Specificity\_ms <- TN/(TN+FP)  
sprintf("Train Specificity: %.3f", round(Tr\_Specificity\_ms, 3))

## [1] "Train Specificity: 0.852"

# Creating Confusion Matrix on Test Dataset  
   
pred.RP\_ms <- predict(RP.mod\_ms, newdata=test\_ms)  
CF\_RP\_MS <- table(Actual=test\_ms$OT\_MS, Predicted=pred.RP\_ms)  
  
CF\_RP\_MS

## Predicted  
## Actual 0 1  
## 0 39 5  
## 1 32 22

TP <- CF\_RP\_MS[2,2]  
TN <- CF\_RP\_MS[1,1]  
FP <- CF\_RP\_MS[1,2]  
FN <- CF\_RP\_MS[2,1]   
  
#Calculate Accuracy  
Tr\_Accuracy\_ms <- ((TP+TN)/sum(CF\_RP\_MS))  
sprintf("Test Accuracy: %.3f", round(Tr\_Accuracy\_ms, 3))

## [1] "Test Accuracy: 0.622"

#Calculate Precision  
Tr\_Precision\_ms <- TP/(TP+FP)  
sprintf("Test Precision: %.3f", round(Tr\_Precision\_ms, 3))

## [1] "Test Precision: 0.815"

#Calculate Sensitivity  
Tr\_Sensitivity\_ms <- TP/(TP+FN)  
sprintf("Test Sensitivity: %.3f", round(Tr\_Sensitivity\_ms, 3))

## [1] "Test Sensitivity: 0.407"

#Calculate Prevalence  
Tr\_Prevalence\_ms <- (TP+FN)/(sum(CF\_RP\_MS))  
sprintf("Test Prevalence: %.3f", round(Tr\_Prevalence\_ms, 3))

## [1] "Test Prevalence: 0.551"

#Calculate Specificity  
Tr\_Specificity\_ms <- TN/(TN+FP)  
sprintf("Test Specificity: %.3f", round(Tr\_Specificity\_ms, 3))

## [1] "Test Specificity: 0.886"

## 3a Neural Network

start\_time <- Sys.time()  
  
set.seed(8430)  
nn.mod\_ms <- nnet(OT\_MS ~ .,  
 data=train\_ms,  
 size=3,  
 rang=0.1,  
 maxit=1200,  
 trace=FALSE)  
  
end\_time <- Sys.time()  
   
NN\_Time\_ms <- end\_time - start\_time  
  
NN\_Time\_ms

## Time difference of 0.009283066 secs

# Creating Confusion Matrix on Train Dataset  
  
pred.nn\_ms <- predict(nn.mod\_ms, newdata=train\_ms, type="class")  
CF\_NN\_MS <- table(Actual=train\_ms$OT\_MS, Predicted=pred.nn\_ms)  
  
CF\_NN\_MS

## Predicted  
## Actual 0 1  
## 0 91 98  
## 1 65 134

TP <- CF\_NN\_MS[2,2]  
TN <- CF\_NN\_MS[1,1]  
FP <- CF\_NN\_MS[1,2]  
FN <- CF\_NN\_MS[2,1]   
  
#Calculate Accuracy  
Tr\_Accuracy\_ms <- ((TP+TN)/sum(CF\_NN\_MS))  
sprintf("Train Accuracy: %.3f", round(Tr\_Accuracy\_ms, 3))

## [1] "Train Accuracy: 0.580"

#Calculate Precision  
Tr\_Precision\_ms <- TP/(TP+FP)  
sprintf("Train Precision: %.3f", round(Tr\_Precision\_ms, 3))

## [1] "Train Precision: 0.578"

#Calculate Sensitivity  
Tr\_Sensitivity\_ms <- TP/(TP+FN)  
sprintf("Train Sensitivity: %.3f", round(Tr\_Sensitivity\_ms, 3))

## [1] "Train Sensitivity: 0.673"

#Calculate Prevalence  
Tr\_Prevalence\_ms <- (TP+FN)/(sum(CF\_NN\_MS))  
sprintf("Train Prevalence: %.3f", round(Tr\_Prevalence\_ms, 3))

## [1] "Train Prevalence: 0.513"

#Calculate Specificity  
Tr\_Specificity\_ms <- TN/(TN+FP)  
sprintf("Train Specificity: %.3f", round(Tr\_Specificity\_ms, 3))

## [1] "Train Specificity: 0.481"

# Creating Confusion Matrix on Test Dataset  
   
pred.nn\_ms <- predict(nn.mod\_ms, newdata=test\_ms, type="class")  
CF\_NN\_MS <- table(Actual=test\_ms$OT\_MS, Predicted=pred.nn\_ms)  
  
CF\_NN\_MS

## Predicted  
## Actual 0 1  
## 0 19 25  
## 1 21 33

TP <- CF\_NN\_MS[2,2]  
TN <- CF\_NN\_MS[1,1]  
FP <- CF\_NN\_MS[1,2]  
FN <- CF\_NN\_MS[2,1]   
  
#Calculate Accuracy  
Tr\_Accuracy\_ms <- ((TP+TN)/sum(CF\_NN\_MS))  
sprintf("Test Accuracy: %.3f", round(Tr\_Accuracy\_ms, 3))

## [1] "Test Accuracy: 0.531"

#Calculate Precision  
Tr\_Precision\_ms <- TP/(TP+FP)  
sprintf("Test Precision: %.3f", round(Tr\_Precision\_ms, 3))

## [1] "Test Precision: 0.569"

#Calculate Sensitivity  
Tr\_Sensitivity\_ms <- TP/(TP+FN)  
sprintf("Test Sensitivity: %.3f", round(Tr\_Sensitivity\_ms, 3))

## [1] "Test Sensitivity: 0.611"

#Calculate Prevalence  
Tr\_Prevalence\_ms <- (TP+FN)/(sum(CF\_NN\_MS))  
sprintf("Test Prevalence: %.3f", round(Tr\_Prevalence\_ms, 3))

## [1] "Test Prevalence: 0.551"

#Calculate Specificity  
Tr\_Specificity\_ms <- TN/(TN+FP)  
sprintf("Test Specificity: %.3f", round(Tr\_Specificity\_ms, 3))

## [1] "Test Specificity: 0.432"

**Interpretation:** ## 4 Comparing all Classifiers Comparing above 4 classifiers based on following measures:

|Logistic | Naïve-Bayes | Recursive Partitioning | Neural Network

Accuracy |0.758 | 0.755 | 0.735 | 0.580  
Consistency |3/5 | 4/5 | 1/5 | 1/5  
Speed |0.08980107| 0.04174495 | 0.05557895 | 0.04200816  
False positive |48 | 45 | 28 | 98

1. Logistic and Naïve-Bayes has similar accuracy and tops other 2.
2. Naïve-Bayes more consistent with very close Accuracy, Precision, Sensitivity and Specificity between Train and Test data sets.
3. Naïve-Bayes is most suitable when processing speed is most important.
4. Recursive Partitioning minimizes false positive most.
5. Recommendation  
   Based on above measures Naïve-Bayes classifier model is superior in terms of Accuracy, Consistency and Speed. I recommend Naïve-Bayes in this case.