BostonHousingExample\_Classification

STAT380

2023-10-12

#Loading the BostonHousing Datya

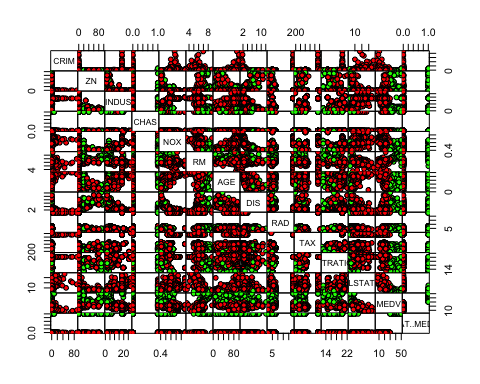
### Example 2: Boston data with binary classifier  
BostonH<-read.csv(url("https://raw.githubusercontent.com/subhadippal2019/STAT380UAEU/main/BostonHousing.csv"))  
set.seed(12)  
attach(BostonH)  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

inTrain = createDataPartition(BostonH$CRIM, p = 0.8, list = FALSE)  
train = BostonH[inTrain, ]  
vali = BostonH[-inTrain,]

# Creating scatter plot for the numerical variables:  
pairs(BostonH,  
 gap = 0,  
 bg = c("red", "green", "blue")[as.factor(CAT..MEDV)],  
 pch = 21)



#Discriminant analysis with one predictor: CRIM  
library(MASS)  
formulas = as.factor(CAT..MEDV) ~ CRIM  
MEDV\_lda.1 = lda(formulas, train)  
p = predict(MEDV\_lda.1, train)  
#ldahist(data = p$x[,1], g = train$CAT..MEDV)  
###Discriminant analysis density plots  
#p.df = data.frame(LD1 = p$x, class = p$class)  
#print(p.df) # Not necessary  
#ggplot(p.df) + geom\_density(aes(LD1, fill = class), alpha = 0.2)

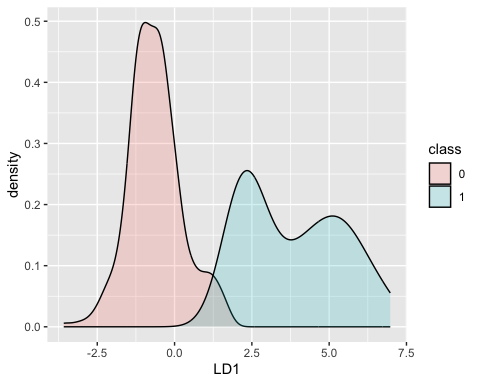
#### Note: Clearly, based on the histograms and density plots, the prediction was not good.  
  
#Discriminant analysis with all predictors  
formulas = as.factor(CAT..MEDV) ~ CRIM + INDUS + NOX + RM + AGE + DIS + PTRATIO + LSTAT + MEDV  
MEDV\_lda = lda(formulas, train)  
MEDV\_lda

## Call:  
## lda(formulas, data = train)  
##   
## Prior probabilities of groups:  
## 0 1   
## 0.8349754 0.1650246   
##   
## Group means:  
## CRIM INDUS NOX RM AGE DIS PTRATIO LSTAT  
## 0 4.1644690 12.292802 0.5647018 6.067617 70.64307 3.702458 18.86460 14.281003  
## 1 0.7836348 5.551493 0.4981940 7.305507 57.77463 4.273193 16.35821 5.148209  
## MEDV  
## 0 19.27404  
## 1 39.17463  
##   
## Coefficients of linear discriminants:  
## LD1  
## CRIM 0.034964242  
## INDUS -0.045230244  
## NOX 1.972884240  
## RM 0.587544779  
## AGE 0.004018909  
## DIS 0.032399356  
## PTRATIO -0.025705505  
## LSTAT 0.072827463  
## MEDV 0.202243942

p = predict(MEDV\_lda, train)  
#ldahist(data = p$x[,1], g = train$CAT..MEDV)  
#ggord(MEDV\_lda, train$CAT..MEDV, ylim =c(-10,10) )  
###Discriminant analysis density plots  
p.df = data.frame(LD1 = p$x, class = p$class)  
print(p.df)

## LD1 class  
## 1 0.223550574 0  
## 2 -0.380832844 0  
## 3 2.273761394 1  
## 4 1.972099516 1  
## 5 2.835025688 1  
## 6 0.904901711 0  
## 8 1.558457816 0  
## 9 -0.095840381 0  
## 10 -0.367260541 0  
## 11 -0.665159889 0  
## 12 -0.668985523 0  
## 13 -0.197902681 0  
## 14 -1.015529854 0  
## 15 -1.144883857 0  
## 16 -1.197077016 0  
## 17 -0.722598963 0  
## 18 -1.040314495 0  
## 19 -1.220585278 0  
## 20 -1.366174777 0  
## 23 -1.074459237 0  
## 25 -1.291408400 0  
## 26 -1.839701689 0  
## 27 -1.271756573 0  
## 28 -1.321936264 0  
## 32 -1.626768485 0  
## 33 -0.970158175 0  
## 35 -1.300157433 0  
## 36 -1.194654951 0  
## 37 -0.925844316 0  
## 38 -0.973134384 0  
## 39 -0.102541974 0  
## 40 1.108216243 0  
## 41 1.995151785 1  
## 42 0.208004136 0  
## 43 -0.322023201 0  
## 44 -0.299761351 0  
## 45 -0.804029244 0  
## 46 -1.410923668 0  
## 47 -0.922725208 0  
## 49 -0.792968164 0  
## 50 -0.854454711 0  
## 51 -0.760526480 0  
## 52 -0.734249771 0  
## 53 -0.061752877 0  
## 54 -0.456273990 0  
## 55 -0.940269393 0  
## 56 2.604352826 1  
## 57 0.124016402 0  
## 59 -0.460917834 0  
## 62 -1.264098990 0  
## 63 -0.375635083 0  
## 64 0.499149291 0  
## 65 2.438734977 1  
## 66 -0.520079600 0  
## 67 -1.185436362 0  
## 70 -1.095577694 0  
## 71 -0.627283354 0  
## 72 -1.124407758 0  
## 73 -1.199692722 0  
## 75 -0.795203433 0  
## 76 -1.010878720 0  
## 77 -0.973276570 0  
## 78 -1.131564207 0  
## 79 -0.785256490 0  
## 81 0.629282660 0  
## 83 -0.168844534 0  
## 84 -0.516656453 0  
## 85 0.030274780 0  
## 87 -0.259668292 0  
## 89 0.288558758 0  
## 91 -0.111473264 0  
## 94 -0.653657127 0  
## 95 -1.008857267 0  
## 96 0.940341381 0  
## 99 4.447305804 1  
## 100 2.359379975 1  
## 101 0.903453684 0  
## 102 0.572262679 0  
## 103 -0.974725355 0  
## 104 -0.778533243 0  
## 105 -0.681534532 0  
## 107 -0.534423446 0  
## 108 -0.545446766 0  
## 109 -0.537686195 0  
## 110 -0.538993019 0  
## 111 -0.425318962 0  
## 112 0.069030527 0  
## 114 -0.566010023 0  
## 116 -0.873587805 0  
## 117 -0.468123852 0  
## 118 -1.048571166 0  
## 119 -0.571824863 0  
## 120 -1.027440039 0  
## 122 -1.235271165 0  
## 124 -1.071262056 0  
## 125 -1.329253113 0  
## 126 -0.969991695 0  
## 127 -1.406301095 0  
## 128 -1.792497147 0  
## 129 -1.111584997 0  
## 130 -2.103177173 0  
## 132 -1.036870716 0  
## 133 -0.423686047 0  
## 134 -1.406931539 0  
## 136 -1.018051701 0  
## 137 -1.426666598 0  
## 138 -1.337811515 0  
## 140 -1.093581742 0  
## 141 -1.461467878 0  
## 142 -1.245797830 0  
## 143 -0.960957780 0  
## 144 -0.476894770 0  
## 146 -0.411311474 0  
## 147 -1.158929262 0  
## 148 -0.847233796 0  
## 149 -0.142135324 0  
## 150 -0.867980219 0  
## 151 0.113853402 0  
## 152 -0.747594117 0  
## 153 -1.992592440 0  
## 154 -0.408222604 0  
## 155 -0.727302389 0  
## 156 -0.984028112 0  
## 159 -0.431746012 0  
## 161 0.123857198 0  
## 163 5.476562121 1  
## 164 5.890827466 1  
## 165 -0.480432898 0  
## 167 5.679112242 1  
## 168 -0.274006190 0  
## 169 -0.015205028 0  
## 171 -1.361605320 0  
## 172 -1.141305772 0  
## 173 0.041902873 0  
## 174 0.209724813 0  
## 175 -0.336430478 0  
## 176 0.998833334 0  
## 178 0.129783122 0  
## 179 1.559182625 0  
## 180 2.899419174 1  
## 181 4.167503487 1  
## 182 2.535328675 1  
## 184 1.902040944 1  
## 185 0.690026399 0  
## 187 5.937815281 1  
## 188 1.735041220 0  
## 189 0.981552953 0  
## 192 1.306945558 0  
## 195 0.805099905 0  
## 196 5.894599677 1  
## 197 2.278463568 1  
## 198 1.906158376 1  
## 199 2.735518822 1  
## 201 1.951585374 1  
## 202 -0.071123355 0  
## 203 4.054194012 1  
## 204 5.508102499 1  
## 205 5.844335082 1  
## 206 -0.761071031 0  
## 207 0.003732497 0  
## 208 -0.101259429 0  
## 209 0.138024953 0  
## 210 -0.397887906 0  
## 211 -0.156817207 0  
## 212 -0.494104373 0  
## 214 0.567461850 0  
## 215 0.483902347 0  
## 216 -0.123734616 0  
## 217 -0.296096757 0  
## 219 -0.154979410 0  
## 220 -0.134194081 0  
## 221 1.110057159 0  
## 222 0.511186267 0  
## 223 1.224745565 0  
## 224 1.439691474 0  
## 225 5.090297793 1  
## 226 6.473622110 1  
## 227 3.473501480 1  
## 228 1.954526278 1  
## 229 4.886379225 1  
## 231 0.140626592 0  
## 232 2.044690229 1  
## 233 4.408933792 1  
## 235 1.262453521 0  
## 236 0.057105426 0  
## 237 0.585201575 0  
## 238 1.936409172 1  
## 239 -0.277773750 0  
## 240 -0.116071907 0  
## 242 -0.599920448 0  
## 243 -0.133103464 0  
## 244 -0.428713318 0  
## 245 -1.391845606 0  
## 246 -0.794558691 0  
## 247 -0.136720133 0  
## 248 -0.590788946 0  
## 249 0.164413525 0  
## 250 0.334032629 0  
## 251 -0.247569018 0  
## 252 -0.377590090 0  
## 253 0.929619160 0  
## 254 4.381406328 1  
## 255 -0.701968243 0  
## 256 -0.897633296 0  
## 257 4.232546528 1  
## 258 6.976265853 1  
## 259 3.591273475 1  
## 260 2.048254997 1  
## 261 3.130633541 1  
## 262 5.059297646 1  
## 263 6.642404052 1  
## 264 2.818030944 1  
## 265 3.603778770 1  
## 268 6.703017113 1  
## 269 4.546227746 1  
## 270 -0.655934715 0  
## 272 -0.234568410 0  
## 273 0.013916413 0  
## 275 1.371761327 0  
## 276 1.354530886 0  
## 277 2.105524025 1  
## 278 1.603475067 0  
## 280 2.249622414 1  
## 281 4.988549448 1  
## 282 2.434116499 1  
## 284 5.860621696 1  
## 285 2.016160892 1  
## 286 -0.336603447 0  
## 287 -0.513446341 0  
## 288 -0.458481769 0  
## 289 -0.486999144 0  
## 290 0.213265679 0  
## 291 0.595667778 0  
## 292 2.561519190 1  
## 293 0.420326612 0  
## 294 -0.679738863 0  
## 295 -0.965430986 0  
## 296 0.493814512 0  
## 297 0.273555075 0  
## 298 -0.889422258 0  
## 299 -0.530028743 0  
## 300 1.135830324 0  
## 301 0.433290889 0  
## 302 -0.294870635 0  
## 304 1.753465753 0  
## 306 1.170997167 0  
## 307 2.522842198 1  
## 308 1.208258901 0  
## 309 -0.365566542 0  
## 311 -2.237790203 0  
## 312 -0.834579257 0  
## 314 -0.588747598 0  
## 315 0.166116756 0  
## 316 -1.747245371 0  
## 317 -0.777343490 0  
## 318 -0.672168327 0  
## 319 -0.087390149 0  
## 321 -0.203008442 0  
## 323 -1.077802336 0  
## 324 -1.267784202 0  
## 325 -0.082542545 0  
## 326 -0.314566900 0  
## 327 -0.569178782 0  
## 328 -0.324650649 0  
## 330 -0.421323175 0  
## 332 -1.611891915 0  
## 334 -0.573901354 0  
## 335 -0.801729856 0  
## 338 -1.155795314 0  
## 339 -0.899561212 0  
## 340 -1.143732823 0  
## 341 -1.194290250 0  
## 343 -1.033371966 0  
## 344 0.220997397 0  
## 345 1.524994806 0  
## 346 -1.329026520 0  
## 347 -1.285666212 0  
## 348 -0.263985938 0  
## 350 0.779076711 0  
## 351 -0.185533794 0  
## 353 -1.340422811 0  
## 354 1.372867756 0  
## 355 -1.624749564 0  
## 357 -0.327476588 0  
## 359 -0.001142007 0  
## 362 -0.327826216 0  
## 363 -0.950710857 0  
## 364 -1.192851054 0  
## 365 0.753514656 0  
## 366 -0.998574105 0  
## 367 -0.817808011 0  
## 368 -1.071585753 0  
## 369 3.978514564 1  
## 370 5.034115238 1  
## 371 5.201880753 1  
## 372 5.313467320 1  
## 373 5.061954745 1  
## 374 -0.800534008 0  
## 375 -0.761536527 0  
## 377 -0.465250145 0  
## 378 -0.815591462 0  
## 379 -0.447024739 0  
## 380 -1.451751068 0  
## 381 1.148318626 0  
## 382 -1.240934037 0  
## 383 -1.740349607 0  
## 384 -1.520890692 0  
## 385 -2.079130055 0  
## 386 -1.942678354 0  
## 387 -1.552675888 0  
## 388 -1.808336591 0  
## 389 -1.656753929 0  
## 390 -2.021791075 0  
## 391 -1.419111638 0  
## 392 0.429004710 0  
## 393 -2.128549683 0  
## 394 -1.518338387 0  
## 395 -1.661656344 0  
## 396 -1.329245991 0  
## 397 -1.437808573 0  
## 398 -2.520150411 0  
## 400 -2.182108686 0  
## 401 -1.856331647 0  
## 403 -1.305789989 0  
## 404 -2.215846184 0  
## 405 -0.975074227 0  
## 406 -0.938520374 0  
## 407 -2.150139457 0  
## 408 0.830063478 0  
## 409 -0.573667497 0  
## 411 -0.586565529 0  
## 412 -0.096668129 0  
## 413 -0.022248963 0  
## 414 -0.731541471 0  
## 415 -0.966129140 0  
## 416 -1.367119157 0  
## 417 -1.630369095 0  
## 418 -1.334570510 0  
## 420 -1.592952002 0  
## 421 -0.648009786 0  
## 424 -1.295029266 0  
## 425 -2.456587843 0  
## 427 -2.632334920 0  
## 428 -1.213493641 0  
## 429 -1.748055194 0  
## 430 -1.614783006 0  
## 431 -1.343845805 0  
## 432 -0.901482829 0  
## 434 -1.267528283 0  
## 435 -1.686956300 0  
## 437 -1.696214970 0  
## 438 -1.397525625 0  
## 439 -1.138468405 0  
## 440 -1.366761631 0  
## 441 -1.337555849 0  
## 442 -0.251159048 0  
## 443 -0.444076250 0  
## 444 -0.580434258 0  
## 449 -1.143655505 0  
## 450 -1.211009726 0  
## 452 -0.809088207 0  
## 453 -0.908481550 0  
## 454 0.183977519 0  
## 455 -0.625020639 0  
## 457 -1.681516998 0  
## 459 -1.150598423 0  
## 460 -0.392405441 0  
## 462 -0.785350360 0  
## 463 -0.416766787 0  
## 464 -0.428743290 0  
## 467 -0.706055927 0  
## 468 -0.440964013 0  
## 469 -0.418963389 0  
## 471 -0.585817137 0  
## 472 -0.935621303 0  
## 473 0.031786351 0  
## 475 -1.955794220 0  
## 476 -1.248366253 0  
## 477 -0.772345788 0  
## 478 -1.599282370 0  
## 479 -1.224516691 0  
## 481 -0.427009248 0  
## 482 -0.171502106 0  
## 483 0.232776894 0  
## 484 -1.156338813 0  
## 485 -1.042790322 0  
## 486 -0.769195918 0  
## 487 -0.821556657 0  
## 488 -1.047656104 0  
## 489 -2.353710992 0  
## 490 -3.583682384 0  
## 491 -3.132157255 0  
## 492 -2.341294829 0  
## 493 -1.423914873 0  
## 494 -0.655243936 0  
## 495 0.092454143 0  
## 496 -0.094531071 0  
## 497 -0.507762508 0  
## 498 -1.073177928 0  
## 500 -1.301820773 0  
## 501 -1.200643492 0  
## 502 -0.295054243 0  
## 503 -0.956225010 0  
## 504 0.017715931 0  
## 505 -0.410257264 0  
## 506 -2.832420084 0

ggplot(p.df) + geom\_density(aes(LD1, fill = class), alpha = 0.2)



####################################################################  
# Classifier Performance  
###################################################################  
  
## Run logit model: Medianvalue0 /1 ~ LowerPopul.+ Numberofrooms+Teacher /Pupilratio  
  
fit = glm(CAT..MEDV~LSTAT+RM+PTRATIO,data=train,family = "binomial")  
summary(fit)

##   
## Call:  
## glm(formula = CAT..MEDV ~ LSTAT + RM + PTRATIO, family = "binomial",   
## data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.0326 -0.1810 -0.0442 -0.0038 3.2965   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -11.7543 5.0132 -2.345 0.019043 \*   
## LSTAT -0.4110 0.1100 -3.736 0.000187 \*\*\*  
## RM 2.8482 0.5976 4.766 1.88e-06 \*\*\*  
## PTRATIO -0.3213 0.1164 -2.761 0.005759 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 363.70 on 405 degrees of freedom  
## Residual deviance: 130.07 on 402 degrees of freedom  
## AIC: 138.07  
##   
## Number of Fisher Scoring iterations: 8

#Create predictions-training vs.validation set  
pred\_t = predict(fit,newdata = train,type ="response")  
pred\_v = predict(fit,newdata = vali,type ="response")  
  
#Evaluate performance-training vs.validation set  
#Training-logit model  
confusionMatrix(as.factor(ifelse(pred\_t > 0.5,1,0)), as.factor(train$CAT..MEDV))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 330 10  
## 1 9 57  
##   
## Accuracy : 0.9532   
## 95% CI : (0.9279, 0.9716)  
## No Information Rate : 0.835   
## P-Value [Acc > NIR] : 1.712e-13   
##   
## Kappa : 0.8292   
##   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.9735   
## Specificity : 0.8507   
## Pos Pred Value : 0.9706   
## Neg Pred Value : 0.8636   
## Prevalence : 0.8350   
## Detection Rate : 0.8128   
## Detection Prevalence : 0.8374   
## Balanced Accuracy : 0.9121   
##   
## 'Positive' Class : 0   
##

#Or  
pred\_t\_c = ifelse(pred\_t > 0.5,1,0); head(pred\_t\_c); length(pred\_t\_c)

## 1 2 3 4 5 6   
## 1 0 1 1 1 0

## [1] 406

confusionMatrix(as.factor(train$CAT..MEDV), as.factor(pred\_t\_c))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 330 9  
## 1 10 57  
##   
## Accuracy : 0.9532   
## 95% CI : (0.9279, 0.9716)  
## No Information Rate : 0.8374   
## P-Value [Acc > NIR] : 4.045e-13   
##   
## Kappa : 0.8292   
##   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.9706   
## Specificity : 0.8636   
## Pos Pred Value : 0.9735   
## Neg Pred Value : 0.8507   
## Prevalence : 0.8374   
## Detection Rate : 0.8128   
## Detection Prevalence : 0.8350   
## Balanced Accuracy : 0.9171   
##   
## 'Positive' Class : 0   
##

#Validation-logit model  
confusionMatrix(as.factor(ifelse(pred\_v>0.5,1,0)), as.factor(vali$CAT..MEDV))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 82 2  
## 1 1 15  
##   
## Accuracy : 0.97   
## 95% CI : (0.9148, 0.9938)  
## No Information Rate : 0.83   
## P-Value [Acc > NIR] : 1.309e-05   
##   
## Kappa : 0.8911   
##   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.9880   
## Specificity : 0.8824   
## Pos Pred Value : 0.9762   
## Neg Pred Value : 0.9375   
## Prevalence : 0.8300   
## Detection Rate : 0.8200   
## Detection Prevalence : 0.8400   
## Balanced Accuracy : 0.9352   
##   
## 'Positive' Class : 0   
##

## Validation  
  
#Naive benchmark:the average  
y\_fit\_naive = median(train$CAT..MEDV)  
#Create predictions  
pred\_v\_reg = predict(fit,newdata = vali,type ="response")  
pred\_v\_naiv = rep(y\_fit\_naive,length(vali$MEDV))  
  
#Evaluate performance-validation set  
#Validation-logit model vs naive benchmark  
confusionMatrix(as.factor(ifelse(pred\_v\_reg > 0.5, 1,0)), as.factor(vali$CAT..MEDV))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 82 2  
## 1 1 15  
##   
## Accuracy : 0.97   
## 95% CI : (0.9148, 0.9938)  
## No Information Rate : 0.83   
## P-Value [Acc > NIR] : 1.309e-05   
##   
## Kappa : 0.8911   
##   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.9880   
## Specificity : 0.8824   
## Pos Pred Value : 0.9762   
## Neg Pred Value : 0.9375   
## Prevalence : 0.8300   
## Detection Rate : 0.8200   
## Detection Prevalence : 0.8400   
## Balanced Accuracy : 0.9352   
##   
## 'Positive' Class : 0   
##

confusionMatrix(as.factor(ifelse(pred\_v\_naiv > 0.5, 1,0)), as.factor(vali$CAT..MEDV))

## Warning in confusionMatrix.default(as.factor(ifelse(pred\_v\_naiv > 0.5, 1, :  
## Levels are not in the same order for reference and data. Refactoring data to  
## match.

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 83 17  
## 1 0 0  
##   
## Accuracy : 0.83   
## 95% CI : (0.7418, 0.8977)  
## No Information Rate : 0.83   
## P-Value [Acc > NIR] : 0.5643017   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 0.0001042   
##   
## Sensitivity : 1.00   
## Specificity : 0.00   
## Pos Pred Value : 0.83   
## Neg Pred Value : NaN   
## Prevalence : 0.83   
## Detection Rate : 0.83   
## Detection Prevalence : 1.00   
## Balanced Accuracy : 0.50   
##   
## 'Positive' Class : 0   
##

## We check the overall classification accuracy   
  
predicted.classes = as.factor(ifelse(pred\_v\_reg > 0.5, 1, 0))  
observed.classes = as.factor(vali$CAT..MEDV)  
#Estimated accuracy-logit model  
accuracy = mean (observed.classes == predicted.classes)  
accuracy

## [1] 0.97

#Estimated miss-classification rate-logit model  
error <-mean (observed.classes != predicted.classes)  
error

## [1] 0.03

#Confusion matrix, proportion of cases-logit model  
table(observed.classes, predicted.classes)

## predicted.classes  
## observed.classes 0 1  
## 0 82 1  
## 1 2 15

prop.table(table(observed.classes, predicted.classes))

## predicted.classes  
## observed.classes 0 1  
## 0 0.82 0.01  
## 1 0.02 0.15

## We check the graphical representation of the logit accuracy  
#Compute the receiver operating characteristics curve (roc)-logit model using library(pROC)  
  
#library(pROC)  
#res.roc = roc(observed.classes, pred\_v\_reg)  
#plot.roc(res.roc, print.auc = TRUE)  
  
#### we repeat the same functions with the Iris data example:  
  
### Confusion matrix and accuracy – training data  
data("iris")  
str(iris)

## 'data.frame': 150 obs. of 5 variables:  
## $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
## $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
## $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
## $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...  
## $ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

head(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 1 5.1 3.5 1.4 0.2 setosa  
## 2 4.9 3.0 1.4 0.2 setosa  
## 3 4.7 3.2 1.3 0.2 setosa  
## 4 4.6 3.1 1.5 0.2 setosa  
## 5 5.0 3.6 1.4 0.2 setosa  
## 6 5.4 3.9 1.7 0.4 setosa

set.seed(134)  
ind = sample(2, nrow(iris), replace = TRUE, prob = c(0.6, 0.4))  
training = iris[ind==1,]  
testing = iris[ind==2,]  
  
iris\_lda = lda(Species~., training)  
  
p1 = predict(iris\_lda, training)$class  
tab = table(Predicted = p1, Actual = training$Species)  
tab

## Actual  
## Predicted setosa versicolor virginica  
## setosa 33 0 0  
## versicolor 0 34 0  
## virginica 0 0 31

p2 = predict(iris\_lda, testing)$class  
tab1 = table(Predicted = p2, Actual = testing$Species)  
tab1

## Actual  
## Predicted setosa versicolor virginica  
## setosa 17 0 0  
## versicolor 0 14 0  
## virginica 0 2 19

n = sum(tab) # number of instances  
nc = nrow(tab) # number of classes  
diag = diag(tab) # number of correctly classified instances per class   
rowsums = apply(tab, 1, sum) # number of instances per class  
colsums = apply(tab, 2, sum) # number of predictions per class  
p = rowsums / n # distribution of instances over the actual classes  
q = colsums / n # distribution of instances over the predicted classes  
  
n = sum(tab1) # number of instances  
nc = nrow(tab1) # number of classes  
diag = diag(tab1) # number of correctly classified instances per class   
rowsums = apply(tab1, 1, sum) # number of instances per class  
colsums = apply(tab1, 2, sum) # number of predictions per class  
p = rowsums / n # distribution of instances over the actual classes  
q = colsums / n # distribution of instances over the predicted classes  
  
accuracy = sum(diag) / n