Упражнение 3

Маркин Артем

17 03 2020

## Постановка задачи

На наборе данных из своего варианта построить указанные модели для прогноза бинарной зависимой переменной. Доля обучающей выборки - 75%.

Построить три графика:

* 1. Матричный график взаимного разброса переменных модели (ggpairs).
  2. Две ROC-кривые на одних осях: сравнение качества прогноза сравниваемых моделей на обучающей выборке.
  3. Две ROC-кривые на одних осях: сравнение качества прогноза сравниваемых моделей на тестовой выборке.

### Вариант 12

* Ядро для set.seed() - 345.
* Данные: Glass{mlbench} - химический состав разных типов стекла.
* Зависимая переменная: Type 1 (1 - наличие признака, все остальные - отсутствие).
* Объясняющие переменные: Все остальные.
* Методы: LDA, QDA.

*Пакеты*:

library('ISLR')  
library('GGally')  
library('MASS')  
library('mlbench')  
  
data(Glass)  
head(Glass)

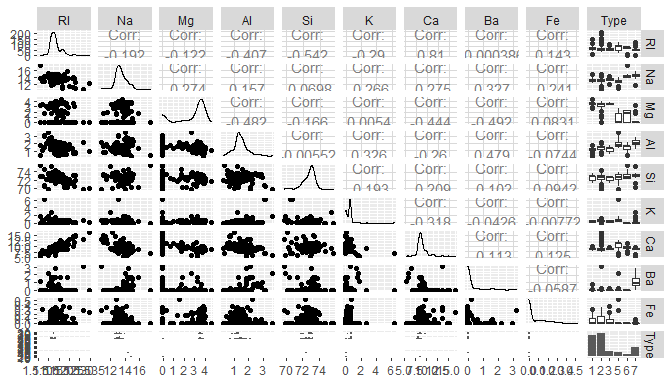
## RI Na Mg Al Si K Ca Ba Fe Type  
## 1 1.52101 13.64 4.49 1.10 71.78 0.06 8.75 0 0.00 1  
## 2 1.51761 13.89 3.60 1.36 72.73 0.48 7.83 0 0.00 1  
## 3 1.51618 13.53 3.55 1.54 72.99 0.39 7.78 0 0.00 1  
## 4 1.51766 13.21 3.69 1.29 72.61 0.57 8.22 0 0.00 1  
## 5 1.51742 13.27 3.62 1.24 73.08 0.55 8.07 0 0.00 1  
## 6 1.51596 12.79 3.61 1.62 72.97 0.64 8.07 0 0.26 1

Зададим ядро генератора случайных чисел и объем обучающей выборки.

# Зададим ядро генератора случайных чисел и объем обучающей выборки  
my.seed <- 345  
train.percent <- 0.75  
options("ggmatrix.progress.bar" = FALSE)

Исходные данные: набор Glass (Химический состав разных типов стекла)

ggp <- ggpairs(Glass)  
print(ggp, progress = FALSE)



# Создаем вектор Type1  
Type1 <- rep(0, length(Glass$Type))  
# Добавляем Type1 во фрейм Glass  
Glass <- cbind(Glass, Type1)  
# Если Type = 1, то Type1 = 1, остальные 0  
for(i in 1:length(Glass$Type)) {if (Glass$Type[i] == 1) {Glass$Type1[i] = 1}}  
  
# Определение долей  
table(Glass$Type1) / sum(table(Glass$Type1))

##   
## 0 1   
## 0.6728972 0.3271028

Для наименьшего класса, в данном случае 0.327, это ошибка нулевого классификатора: если бы мы прогнозировали Type = 1 для всех наблюдений, ровно в такой доле случаев мы бы ошиблись. Точность моделей целесообразно будет сравнивать с этой величиной.

# Отбираем наблюдения в обучающую выборку  
set.seed(my.seed)  
inTrain <- sample(seq\_along(Glass$Type1),  
 nrow(Glass)\*train.percent)  
df <- Glass[inTrain, ]  
dfp <- Glass[-inTrain, ]  
  
# Фактические значения на обучающей выборке  
Fact <- df$Type1  
# Фактические значения на тестовой выборке  
Factp <- dfp$Type1

# Строим модели, чтобы спрогнозировать Type

#LDA

model.lda <- lda(Type1 ~ RI + Na + Mg + Al + Si + K + Ca + Ba + Fe, data = Glass[inTrain, ])  
  
model.lda

## Call:  
## lda(Type1 ~ RI + Na + Mg + Al + Si + K + Ca + Ba + Fe, data = Glass[inTrain,   
## ])  
##   
## Prior probabilities of groups:  
## 0 1   
## 0.66875 0.33125   
##   
## Group means:  
## RI Na Mg Al Si K Ca Ba  
## 0 1.518115 13.50262 2.241402 1.589533 72.66607 0.5336449 9.012710 0.25934579  
## 1 1.518708 13.26170 3.541698 1.192830 72.61679 0.4571698 8.754151 0.01679245  
## Fe  
## 0 0.06056075  
## 1 0.05037736  
##   
## Coefficients of linear discriminants:  
## LD1  
## RI 279.0207631  
## Na 3.9436192  
## Mg 4.5385250  
## Al 3.1076452  
## Si 4.5992250  
## K 4.5123681  
## Ca 3.6573178  
## Ba 3.9965381  
## Fe -0.6147707

# Прогноз: вероятности принадлежности классу Type = 1  
  
p.lda <- predict(model.lda, df, type = 'response')  
  
Forecast1 <- factor(ifelse(p.lda$posterior[, '1'] > 0.5, 2, 1), levels = c(1, 2), labels = c('0', '1'))  
  
#Матрица неточностей  
conf.m <- table(Fact, Forecast1)  
conf.m

## Forecast1  
## Fact 0 1  
## 0 95 12  
## 1 17 36

# Чувствительность  
conf.m[2, 2] / sum(conf.m[2, ])

## [1] 0.6792453

# Специфичность  
conf.m[1, 1] / sum(conf.m[1, ])

## [1] 0.8878505

# Верность  
sum(diag(conf.m)) / sum(conf.m)

## [1] 0.81875

# Ошибка нулевого классификатора  
sum(Glass$Type1 == 1) / length(Glass$Type1)

## [1] 0.3271028

У этой модели сильная чувствительность

# QDA

model.qda <- qda(Type1 ~ RI + Na + Mg + Al + Si + K + Ca + Ba + Fe, data = Glass[inTrain, ])  
model.qda

## Call:  
## qda(Type1 ~ RI + Na + Mg + Al + Si + K + Ca + Ba + Fe, data = Glass[inTrain,   
## ])  
##   
## Prior probabilities of groups:  
## 0 1   
## 0.66875 0.33125   
##   
## Group means:  
## RI Na Mg Al Si K Ca Ba  
## 0 1.518115 13.50262 2.241402 1.589533 72.66607 0.5336449 9.012710 0.25934579  
## 1 1.518708 13.26170 3.541698 1.192830 72.61679 0.4571698 8.754151 0.01679245  
## Fe  
## 0 0.06056075  
## 1 0.05037736

# Прогноз: вероятности принадлежности классу Type = 1  
p.qda <- predict(model.qda, df, type = 'response')  
  
Forecast2 <- factor(ifelse(p.qda$posterior[, '1'] > 0.5, 2, 1), levels = c(1, 2), labels = c('0', '1'))  
  
# матрица неточностей  
conf.m <- table(Fact, Forecast2)  
conf.m

## Forecast2  
## Fact 0 1  
## 0 59 48  
## 1 1 52

# Чувствительность  
conf.m[2, 2] / sum(conf.m[2, ])

## [1] 0.9811321

# Специфичность  
conf.m[1, 1] / sum(conf.m[1, ])

## [1] 0.5514019

# Верность  
sum(diag(conf.m)) / sum(conf.m)

## [1] 0.69375

# Ошибка нулевого классификатора  
sum(Glass$Type1 == 1) / length(Glass$Type1)

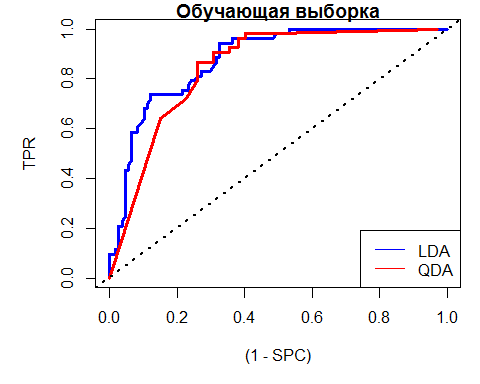
## [1] 0.3271028

У этой модели очень сильная чувствительность

# Подбор границы отсечения вероятностей классов

# ROC-кривые для обучающей выборки

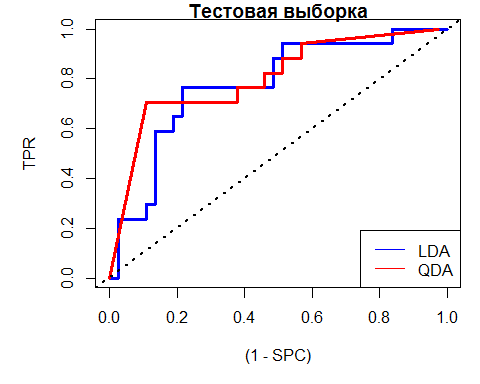
# Считаем 1-SPC и TPR для всех вариантов границы отсечения  
# Для (1 - SPC)  
x1 <- NULL   
# Для TPR  
y1 <- NULL  
  
# LDA  
# Заготовка под матрицу неточностей  
tbl1 <- as.data.frame(matrix(rep(0, 4), 2, 2))  
rownames(tbl1) <- c('fact.0', 'fact.1')  
colnames(tbl1) <- c('predict.0', 'predict.1')  
  
# Цикл по вероятностям отсечения  
for(p in seq(0, 1, length = 501)){  
 # Прогноз  
 Forecast1 <- factor(ifelse(p.lda$posterior[, '1'] > p, 2, 1), levels = c(1, 2), labels = c('0', '1'))  
   
 # Фрейм со сравнением факта и прогноза  
 df.compare <- data.frame(Fact = Fact, Forecast = Forecast1)  
   
 #Заполняем матрицу неточностей  
 tbl1[1, 1] <- nrow(df.compare[df.compare$Fact == '0' & df.compare$Forecast == '0', ])  
 tbl1[2, 2] <- nrow(df.compare[df.compare$Fact == '1' & df.compare$Forecast == '1', ])  
 tbl1[1, 2] <- nrow(df.compare[df.compare$Fact == '0' & df.compare$Forecast == '1', ])  
 tbl1[2, 1] <- nrow(df.compare[df.compare$Fact == '1' & df.compare$Forecast == '0', ])  
   
 # Считаем характиристики  
 TPR <- tbl1[2, 2] / sum(tbl1[2, ])  
 y1 <- c(y1, TPR)  
 SPC <- tbl1[1, 1] / sum(tbl1[1, ])  
 x1 <- c(x1, 1 - SPC)  
}  
  
# QDA  
# Для (1 - SPC)  
x2 <- NULL  
# Для TPR  
y2 <- NULL  
# Заготовка под матрицу неточностей  
tbl2 <- as.data.frame(matrix(rep(0, 4), 2, 2))  
rownames(tbl2) <- c('fact.0', 'fact.1')  
colnames(tbl2) <- c('predict.0', 'predict.1')  
# Цикл по вероятностям отсечения  
for (p in seq(0, 1, length = 501)){  
 # Прогноз  
 Forecast2 <- factor(ifelse(p.qda$posterior[, '1'] > p, 2, 1),  
 levels = c(1, 2),  
 labels = c('0', '1'))  
   
 # фрейм со сравнением факта и прогноза  
 df.compare <- data.frame(Fact = Fact, Forecast = Forecast2)  
   
 # Заполняем матрицу неточностей  
 tbl2[1, 1] <- nrow(df.compare[df.compare$Fact == '0' & df.compare$Forecast == '0', ])  
 tbl2[2, 2] <- nrow(df.compare[df.compare$Fact == '1' & df.compare$Forecast == '1', ])  
 tbl2[1, 2] <- nrow(df.compare[df.compare$Fact == '0' & df.compare$Forecast == '1', ])  
 tbl2[2, 1] <- nrow(df.compare[df.compare$Fact == '1' & df.compare$Forecast == '0', ])  
   
 # Считаем характеристики  
 TPR <- tbl2[2, 2] / sum(tbl2[2, ])  
 y2 <- c(y2, TPR)  
 SPC <- tbl2[1, 1] / sum(tbl2[1, ])  
 x2 <- c(x2, 1 - SPC)  
}  
  
# Строим ROC-кривую  
par(mar = c(5, 5, 1, 1))  
  
# Кривая (логистическая регрессия)  
plot(x1, y1, type = 'l', col = 'blue', lwd = 3,  
 xlab = '(1 - SPC)', ylab = 'TPR',  
 xlim = c(0, 1), ylim = c(0, 1), main = 'Обучающая выборка')  
  
# Кривая (LDA)  
lines(x2, y2, type = 'l', col = 'red', lwd = 3)  
  
# Прямая случайного классификатора  
abline(a = 0, b = 1, lty = 3, lwd = 2)  
  
# Легенда  
legend('bottomright', names <- c('LDA', 'QDA'), lty = 1, col = c('blue', 'red'))



Сравнивая ROC-кривые, полученные на обучающей выборке, сложно сказать, какая из моделей наиболее предпочтительна. Для того, чтобы ответить на этот вопрос построим ROC-кривые на тестовых данных.

# ROC-кривые для тестовой выборки

# LDA  
# Прогноз: вероятности принадлежности классу Type = 1  
p.lda <- predict(model.lda, dfp,   
 type = 'response')  
# Считаем 1-SPC и TPR для всех вариантов границы отсечения  
x1 <- NULL # Для (1 - SPC)  
y1 <- NULL # Для TPR  
# Заготовка под матрицу неточностей  
tbl1 <- as.data.frame(matrix(rep(0, 4), 2, 2))  
rownames(tbl1) <- c('fact.0', 'fact.1')  
colnames(tbl1) <- c('predict.0', 'predict.1')  
# Цикл по вероятностям отсечения  
for (p in seq(0, 1, length = 501)){  
 # Прогноз  
 Forecast1 <- factor(ifelse(p.lda$posterior[, '1'] > p, 2, 1),  
 levels = c(1, 2),  
 labels = c('0', '1'))  
 # Фрейм со сравнением факта и прогноза  
 df.compare <- data.frame(Fact = Factp, Forecast = Forecast1)  
 # Заполняем матрицу неточностей  
 tbl1[1, 1] <- nrow(df.compare[df.compare$Fact == '0' & df.compare$Forecast == '0', ])  
 tbl1[2, 2] <- nrow(df.compare[df.compare$Fact == '1' & df.compare$Forecast == '1', ])  
 tbl1[1, 2] <- nrow(df.compare[df.compare$Fact == '0' & df.compare$Forecast == '1', ])  
 tbl1[2, 1] <- nrow(df.compare[df.compare$Fact == '1' & df.compare$Forecast == '0', ])  
 # Считаем характеристики  
 TPR <- tbl1[2, 2] / sum(tbl1[2, ])  
 y1 <- c(y1, TPR)  
 SPC <- tbl1[1, 1] / sum(tbl1[1, ])  
 x1 <- c(x1, 1 - SPC)}  
# QDA  
# Прогноз: вероятности принадлежности классу Type = 1  
p.qda <- predict(model.qda, dfp,   
 type = 'response')  
x2 <- NULL # для (1 - SPC)  
y2 <- NULL # для TPR  
# Заготовка под матрицу неточностей  
tbl2 <- as.data.frame(matrix(rep(0, 4), 2, 2))  
rownames(tbl2) <- c('fact.0', 'fact.1')  
colnames(tbl2) <- c('predict.0', 'predict.1')  
# Цикл по вероятностям отсечения  
for (p in seq(0, 1, length = 501)){  
 # Прогноз  
 Forecast2 <- factor(ifelse(p.qda$posterior[, '1'] > p, 2, 1),  
 levels = c(1, 2),  
 labels = c('0', '1'))  
   
 # Фрейм со сравнением факта и прогноза  
 df.compare <- data.frame(Fact = Factp, Forecast = Forecast2)  
   
 # Заполняем матрицу неточностей  
 tbl2[1, 1] <- nrow(df.compare[df.compare$Fact == '0' & df.compare$Forecast == '0', ])  
 tbl2[2, 2] <- nrow(df.compare[df.compare$Fact == '1' & df.compare$Forecast == '1', ])  
 tbl2[1, 2] <- nrow(df.compare[df.compare$Fact == '0' & df.compare$Forecast == '1', ])  
 tbl2[2, 1] <- nrow(df.compare[df.compare$Fact == '1' & df.compare$Forecast == '0', ])  
   
 # Считаем характеристики  
 TPR <- tbl2[2, 2] / sum(tbl2[2, ])  
 y2 <- c(y2, TPR)  
 SPC <- tbl2[1, 1] / sum(tbl2[1, ])  
 x2 <- c(x2, 1 - SPC)  
   
}  
# Строим ROC-кривую  
par(mar = c(5, 5, 1, 1))  
# Кривая (логистическая регрессия)  
plot(x1, y1, type = 'l', col = 'blue', lwd = 3,  
 xlab = '(1 - SPC)', ylab = 'TPR',   
 xlim = c(0, 1), ylim = c(0, 1), main = 'Тестовая выборка')  
# Кривая (LDA)  
lines(x2, y2, type = 'l', col = 'red', lwd = 3)  
# Прямая случайного классификатора  
abline(a = 0, b = 1, lty = 3, lwd = 2)  
# Легенда  
legend('bottomright', names <- c('LDA', 'QDA'), lty = 1, col = c('blue', 'red'))



Сравнивая ROC-кривые, полученные на тестовой выборке, видно, что LDA-модель обладает большей предсказательной способностью, чем QDA.

Линейный дискриминантный анализ не имеет столько допущений, как квадратичный дискриминантный анализ. Поэтому если допущения квадратичного дискриминантного анализа не выполняются, то линейный дискриминантный анализ является лучшем средством для анализа.