Упражнение 2

Дроздецкая Анна

08 03 2020

## Постановка задачи

1. Построить модели на данных примера 3 с параметрами распределений соответствующими своему варианту. На графиках сетку с итинной разделябщей границей рисовать не нужно. Определить, какой из методов срабатывает на этих данных лучше и почему.
2. По матрице неточностей той модели, которая оказалась лучше по Acc, рассчитать характеристики качества и ошибки из лекции: , , , , , , , .

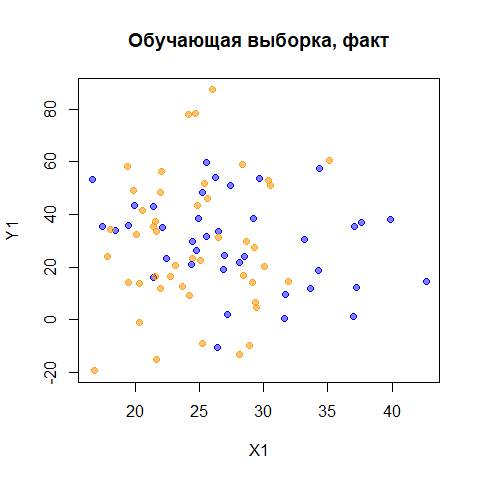
Вариант 6.

* класс :
* класс :

library('knitr')  
library('class')  
library('e1071')  
library('MASS')  
  
# Ядро  
my.seed <- 12345  
  
# Генерируем данные  
n <- 100  
train.percent <- 0.85  
  
# х-ы -- двумерные нормальные случайные величины  
set.seed(my.seed)  
class.0 <- mvrnorm(45, mu = c(30, 26),  
 Sigma = matrix(c(5.4^2, 0, 0, 15.4^2), 2, 2,  
 byrow = T))  
set.seed(my.seed + 1)  
class.1 <- mvrnorm(55, mu = c(25, 29),  
 Sigma = matrix(c(3.6^2, 0, 0, 22.6^2), 2, 2,  
 byrow = T))  
# Записываем х-ы в единые векторы (объединяем классы 0 и 1)  
x1 <- c(class.0[, 1], class.1[, 1])  
x2 <- c(class.0[, 2], class.1[, 2])  
  
# Фактические классы Y  
y <- c(rep(0, nrow(class.0)), rep(1, nrow(class.1)))  
  
# Классы для наблюдений сетки  
rules <- function(x1, x2){  
 ifelse(x2 < 1.6\*x1 + 19, 0, 1)  
}  
  
# Отбираем наблюдения в обучающую выборку  
  
set.seed(my.seed)  
inTrain <- sample(seq\_along(x1), train.percent\*n)  
x1.train <- x1[inTrain]  
x2.train <- x2[inTrain]  
x1.test <- x1[-inTrain]  
x2.test <- x2[-inTrain]  
  
# используем истинные правила, чтобы присвоить фактические классы  
y.train <- y[inTrain]  
y.test <- y[-inTrain]  
  
# фрейм с обучающей выборкой  
df.train.1 <- data.frame(x1 = x1.train, x2 = x2.train, y = y.train)  
# фрейм с тестовой выборкой  
df.test.1 <- data.frame(x1 = x1.test, x2 = x2.test)

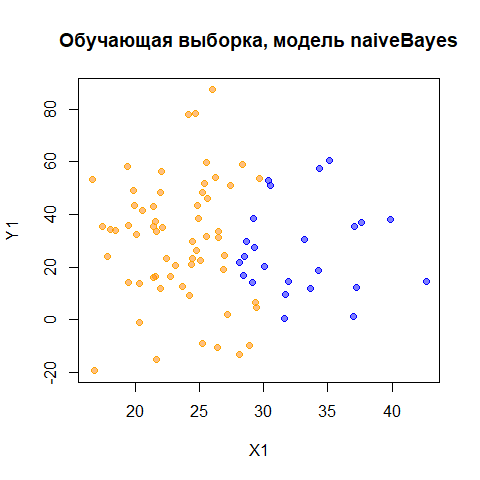
Нарисуем обучающую выборку на графике. Сеткой точек показаны области классов, соответствующие истинным дискриминирующим правилам.

# Рисуем обучающую выборку на графике  
# цвета для графиков  
cls <- c('blue', 'orange')  
cls.t <- c(rgb(0, 0, 1, alpha = 0.5), rgb(1,0.5,0, alpha = 0.5))  
  
# график истинных классов  
plot(df.train.1$x1, df.train.1$x2,   
 pch = 21, bg = cls.t[df.train.1[, 'y'] + 1],   
 col = cls[df.train.1[, 'y'] + 1],  
 xlab = 'X1', ylab = 'Y1',  
 main = 'Обучающая выборка, факт')



Обучим модель **наивного байесовского классификатора** и оценим её точность (верность) на обучающей выборке. Поскольку объясняющие переменные для классов сгенерированы как двумерные нормальные распределения и сами классы не перекрываются, следует ожидать, что эта модель окажется точной.

# Байесовский классификатор  
# наивный байес: непрерывные объясняющие переменные  
  
# строим модель  
nb <- naiveBayes(y ~ ., data = df.train.1)  
# получаем модельные значения на обучающей выборке как классы  
y.nb.train <- ifelse(predict(nb, df.train.1[, -3],   
 type = "raw")[, 2] > 0.5, 1, 0)  
  
# точки наблюдений, предсказанных по модели  
plot(df.train.1$x1, df.train.1$x2,   
 pch = 21, bg = cls.t[y.nb.train + 1],  
 col = cls[y.nb.train + 1],   
 xlab = 'X1', ylab = 'Y1',  
 main = 'Обучающая выборка, модель naiveBayes')



# матрица неточностей на обучающей выборке  
tbl1 <- table(y.train, y.nb.train)  
tbl1

## y.nb.train  
## y.train 0 1  
## 0 15 24  
## 1 9 37

# точность, или верность (Accuracy)  
Acc1 <- sum(diag(tbl1)) / sum(tbl1)  
Acc1

## [1] 0.6117647

Сделаем прогноз классов Y на тестовую выборку и оценим точность модели.

# прогноз на тестовую выборку  
y.nb.test <- ifelse(predict(nb, df.test.1, type = "raw")[, 2] > 0.5, 1, 0)  
  
# матрица неточностей на тестовой выборке  
tbl1 <- table(y.test, y.nb.test)  
tbl1

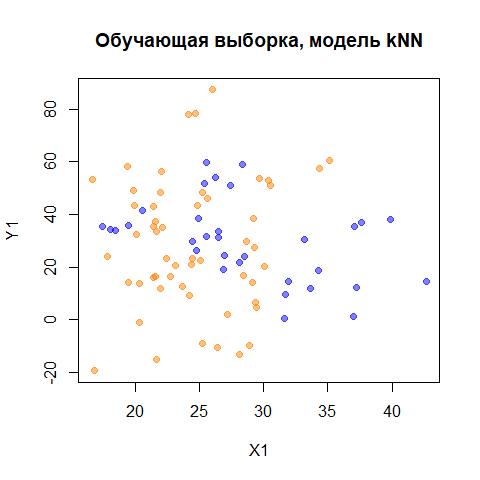
## y.nb.test  
## y.test 0 1  
## 0 3 3  
## 1 2 7

# точность, или верность (Accuracy)  
Acc1 <- sum(diag(tbl1)) / sum(tbl1)  
Acc1

## [1] 0.6666667

Построим модель **kNN**. С этими данными у метода не должно возникнуть проблем, так как классы не смешиваются.

# Метод kNN  
# k = 3  
  
# строим модель и делаем прогноз  
y.knn.train <- knn(train = scale(df.train.1[, -3]),   
 test = scale(df.train.1[, -3]),  
 cl = df.train.1$y, k = 3)  
  
# точки наблюдений, предсказанных по модели  
plot(df.train.1$x1, df.train.1$x2,   
 pch = 21, bg = cls.t[as.numeric(y.knn.train)],   
 col = cls.t[as.numeric(y.knn.train)],  
 xlab = 'X1', ylab = 'Y1',  
 main = 'Обучающая выборка, модель kNN')



# матрица неточностей на обучающей выборке  
tbl2 <- table(y.train, y.knn.train)  
tbl2

## y.knn.train  
## y.train 0 1  
## 0 26 13  
## 1 6 40

# точность (Accuracy)  
Acc2 <- sum(diag(tbl2)) / sum(tbl2)  
Acc2

## [1] 0.7764706

Сделаем прогноз классов Y на тестовую выборку и оценим точность модели.

# прогноз на тестовую выборку  
y.knn.test <- knn(train = scale(df.train.1[, -3]),   
 test = scale(df.test.1[, -3]),  
 cl = df.train.1$y, k = 3)  
  
# матрица неточностей на тестовой выборке  
tbl2 <- table(y.test, y.knn.test)  
tbl2

## y.knn.test  
## y.test 0 1  
## 0 5 1  
## 1 3 6

# точность (Accuracy)  
Acc2 <- sum(diag(tbl2)) / sum(tbl2)  
Acc2

## [1] 0.7333333

Так как значения Acc по тестовой выборке оказались лучше у второй модели (knn), рассчитаем для нее характеристики качества.

TPR <- round(tbl2[2,2]/sum(tbl2[2,]),3) # чувствительность  
SPC <- round(tbl2[1,1]/sum(tbl2[,1]),3) # специфичность  
PPV <- round(tbl2[2,2]/sum(tbl2[,2]),3) # ценность положительного прогноза  
NPV <- round(tbl2[1,1]/sum(tbl2[,1]),3) # ценность отрицательного прогноза  
FNR <- 1-TPR # доля положительных исходов  
FPR <- 1-SPC # доля ложных срабатываний  
FDR <- 1-PPV # доля ложного обнаружения  
MCC <- round((tbl2[1,1]\*tbl2[2,2]-tbl2[1,2]\*tbl2[2,1])/sqrt(sum(tbl2[,2])\*sum(tbl2[2,])\*sum(tbl2[1,])\*sum(tbl2[,1])),3) # корреляция Мэтьюса   
  
ch.fr <- rbind(TPR, SPC, PPV, NPV, FNR, FPR, FDR, MCC)  
colnames(ch.fr) <- 'Модель'  
kable(ch.fr)

|  |  |
| --- | --- |
|  | Модель |
| TPR | 0.667 |
| SPC | 0.625 |
| PPV | 0.857 |
| NPV | 0.625 |
| FNR | 0.333 |
| FPR | 0.375 |
| FDR | 0.143 |
| MCC | 0.491 |