|  |
| --- |
| > library(ISLR)  > Carseats$HighSales <- ifelse(test = Carseats$Sales > 9.32,yes = 'Yes',no = 'No')  > Carseats$HighSales <- as.factor(Carseats$HighSales)  > Carseats <- Carseats[,-1]  > str(Carseats)  'data.frame': 400 obs. of 11 variables:  $ CompPrice : num 138 111 113 117 141 124 115 136 132 132 ...  $ Income : num 73 48 35 100 64 113 105 81 110 113 ...  $ Advertising: num 11 16 10 4 3 13 0 15 0 0 ...  $ Population : num 276 260 269 466 340 501 45 425 108 131 ...  $ Price : num 120 83 80 97 128 72 108 120 124 124 ...  $ ShelveLoc : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...  $ Age : num 42 65 59 55 38 78 71 67 76 76 ...  $ Education : num 17 10 12 14 13 16 15 10 10 17 ...  $ Urban : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...  $ US : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...  $ HighSales : Factor w/ 2 levels "No","Yes": 2 2 2 1 1 2 1 2 1 1 ...  > num.vars <- c(1:5,7,8)  > apply(X = Carseats[,num.vars], MARGIN = 2, FUN = shapiro.test)  $CompPrice  Shapiro-Wilk normality test  data: newX[, i]  W = 0.99843, p-value = 0.9772  $Income  Shapiro-Wilk normality test  data: newX[, i]  W = 0.9611, p-value = 8.396e-09  $Advertising  Shapiro-Wilk normality test  data: newX[, i]  W = 0.87354, p-value < 2.2e-16  $Population  Shapiro-Wilk normality test  data: newX[, i]  W = 0.95201, p-value = 4.081e-10  $Price  Shapiro-Wilk normality test  data: newX[, i]  W = 0.99592, p-value = 0.3902  $Age  Shapiro-Wilk normality test  data: newX[, i]  W = 0.95672, p-value = 1.865e-09  $Education  Shapiro-Wilk normality test  data: newX[, i]  W = 0.9242, p-value = 2.427e-13  > library(bnlearn)  Attaching package: ‘bnlearn’  The following object is masked from ‘package:stats’:  sigma  > to.discretize <- c("Education", "Age", "Population", "Advertising", "Income")  > summary(Carseats$Advertising)  Min. 1st Qu. Median Mean 3rd Qu. Max.  0.000 0.000 5.000 6.635 12.000 29.000  > library(ggplot2)  > ggplot(data = Carseats, mapping = aes(x = Advertising)) +  + geom\_histogram(bins = 30)  > discretized <- discretize(data = Carseats[,to.discretize],  + method = 'quantile',  + breaks = c(5,5,5,2,5))  > summary(discretized)  Education Age Population Advertising Income  [10,11]:96 [25,36] :82 [10,110] :80 [0,5] :206 [21,39] :82  (11,13]:92 (36,48.6]:78 (110,219]:80 (5,29]:194 (39,62] :82  (13,15]:76 (48.6,60]:85 (219,318]:80 (62,77] :77  (15,17]:96 (60,70] :78 (318,412]:80 (77,96.2] :79  (17,18]:40 (70,80] :77 (412,509]:80 (96.2,120]:80  > cols.to.add <- setdiff(names(Carseats), names(discretized))  > carseats.new <- data.frame(cbind(Carseats[,cols.to.add], discretized))  > str(carseats.new)  'data.frame': 400 obs. of 11 variables:  $ CompPrice : num 138 111 113 117 141 124 115 136 132 132 ...  $ Price : num 120 83 80 97 128 72 108 120 124 124 ...  $ ShelveLoc : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...  $ Urban : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...  $ US : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...  $ HighSales : Factor w/ 2 levels "No","Yes": 2 2 2 1 1 2 1 2 1 1 ...  $ Education : Factor w/ 5 levels "[10,11]","(11,13]",..: 4 1 2 3 2 4 3 1 1 4 ...  $ Age : Factor w/ 5 levels "[25,36]","(36,48.6]",..: 2 4 3 3 2 5 5 4 5 5 ...  $ Population : Factor w/ 5 levels "[10,110]","(110,219]",..: 3 3 3 5 4 5 1 5 1 2 ...  $ Advertising: Factor w/ 2 levels "[0,5]","(5,29]": 2 2 2 1 1 2 1 2 1 1 ...  $ Income : Factor w/ 5 levels "[21,39]","(39,62]",..: 3 2 1 5 3 5 5 4 5 5 ...  > carseats.new <- carseats.new[,names(Carseats)]  > str(carseats.new)  'data.frame': 400 obs. of 11 variables:  $ CompPrice : num 138 111 113 117 141 124 115 136 132 132 ...  $ Income : Factor w/ 5 levels "[21,39]","(39,62]",..: 3 2 1 5 3 5 5 4 5 5 ...  $ Advertising: Factor w/ 2 levels "[0,5]","(5,29]": 2 2 2 1 1 2 1 2 1 1 ...  $ Population : Factor w/ 5 levels "[10,110]","(110,219]",..: 3 3 3 5 4 5 1 5 1 2 ...  $ Price : num 120 83 80 97 128 72 108 120 124 124 ...  $ ShelveLoc : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...  $ Age : Factor w/ 5 levels "[25,36]","(36,48.6]",..: 2 4 3 3 2 5 5 4 5 5 ...  $ Education : Factor w/ 5 levels "[10,11]","(11,13]",..: 4 1 2 3 2 4 3 1 1 4 ...  $ Urban : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...  $ US : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...  $ HighSales : Factor w/ 2 levels "No","Yes": 2 2 2 1 1 2 1 2 1 1 ...  > library(caret)  Loading required package: lattice  > library(lattice)  > library(caret)  > set.seed(1010)  > train.indices <- createDataPartition(carseats.new$HighSales, p = 0.8, list = FALSE)  > train.data <- carseats.new[train.indices,]  > test.data <- carseats.new[-train.indices,]  > library(e1071)  Attaching package: ‘e1071’  The following object is masked from ‘package:bnlearn’:  impute  > nb1 <- naiveBayes(HighSales ~ ., data = train.data)  > print(nb1)  Naive Bayes Classifier for Discrete Predictors  Call:  naiveBayes.default(x = X, y = Y, laplace = laplace)  A-priori probabilities:  Y  No Yes  0.7507788 0.2492212  Conditional probabilities:  CompPrice  Y [,1] [,2]  No 124.7967 14.86117  Yes 125.6500 17.22164  Income  Y [21,39] (39,62] (62,77] (77,96.2] (96.2,120]  No 0.2406639 0.2323651 0.1701245 0.1784232 0.1784232  Yes 0.1125000 0.1875000 0.2375000 0.1750000 0.2875000  Advertising  Y [0,5] (5,29]  No 0.5767635 0.4232365  Yes 0.2875000 0.7125000  Population  Y [10,110] (110,219] (219,318] (318,412] (412,509]  No 0.1991701 0.2240664 0.1867220 0.2033195 0.1867220  Yes 0.1625000 0.1625000 0.2375000 0.2375000 0.2000000  Price  Y [,1] [,2]  No 119.6805 22.24320  Yes 103.6625 23.85608  ShelveLoc  Y Bad Good Medium  No 0.3195021 0.1078838 0.5726141  Yes 0.0625000 0.5375000 0.4000000  Age  Y [25,36] (36,48.6] (48.6,60] (60,70] (70,80]  No 0.1908714 0.1742739 0.1991701 0.2033195 0.2323651  Yes 0.3000000 0.2375000 0.2125000 0.1500000 0.1000000  Education  Y [10,11] (11,13] (13,15] (15,17] (17,18]  No 0.22821577 0.22406639 0.21161826 0.24896266 0.08713693  Yes 0.27500000 0.17500000 0.20000000 0.27500000 0.07500000  Urban  Y No Yes  No 0.2946058 0.7053942  Yes 0.3375000 0.6625000  US  Y No Yes  No 0.4107884 0.5892116  Yes 0.1625000 0.8375000  > nb1.pred <- predict(nb1, newdata = test.data, type = 'class')  > head(nb1.pred)  [1] No Yes No Yes No Yes  Levels: No Yes  > nb1.cm <- table(true = test.data$HighSales, predicted = nb1.pred)  > nb1.cm  predicted  true No Yes  No 58 2  Yes 4 15  > compute.eval.metrics <- function(cmatrix) {  + TP <- cmatrix[1,1] # true positive  + TN <- cmatrix[2,2] # true negative  + FP <- cmatrix[2,1] # false positive  + FN <- cmatrix[1,2] # false negative  + acc = sum(diag(cmatrix)) / sum(cmatrix)  + precision <- TP / (TP + FP)  + recall <- TP / (TP + FN)  + F1 <- 2\*precision\*recall / (precision + recall)  + c(accuracy = acc, precision = precision, recall = recall, F1 = F1)  + }  > # source("EvaluationMeasures.R")  > # compute the evaluation metrics  > nb1.eval <- compute.eval.metrics(nb1.cm)  > nb1.eval  accuracy precision recall F1  0.9240506 0.9354839 0.9666667 0.9508197  > nb2 <- naiveBayes(HighSales ~ ShelveLoc + Price + Advertising + Age + CompPrice,  + data = train.data)  > nb2.pred <- predict(nb2, newdata = test.data, type = 'class')  > nb2.cm <- table(true = test.data$HighSales, predicted = nb2.pred)  > nb2.cm  predicted  true No Yes  No 57 3  Yes 7 12  > nb2.eval <- compute.eval.metrics(nb2.cm)  > nb2.eval  accuracy precision recall F1  0.8734177 0.8906250 0.9500000 0.9193548  > data.frame(rbind(nb1.eval, nb2.eval), row.names = c("NB\_mod 1", "NB\_mod 2"))  accuracy precision recall F1  NB\_mod 1 0.9240506 0.9354839 0.9666667 0.9508197  NB\_mod 2 0.8734177 0.8906250 0.9500000 0.9193548  > nb2.pred.prob <- predict(nb2, newdata = test.data, type = "raw")  > # note that the type parameter is now set to 'raw'  > head(nb2.pred.prob)  No Yes  [1,] 0.9556750 0.04432504  [2,] 0.4081647 0.59183527  [3,] 0.9710644 0.02893557  [4,] 0.4377776 0.56222238  [5,] 0.9655643 0.03443575  [6,] 0.1455803 0.85441974  > library(pROC)  Type 'citation("pROC")' for a citation.  Attaching package: ‘pROC’  The following objects are masked from ‘package:stats’:  cov, smooth, var  > nb2.roc <- roc(response = as.numeric(test.data$HighSales),  + predictor = nb2.pred.prob[,1],  + levels = c(2, 1))  Setting direction: controls < cases  > nb2.roc$auc  Area under the curve: 0.9684  > plot.roc(nb2.roc,  + print.thres = TRUE,  + print.thres.best.method = "youden")  > nb2.coords <- coords(nb2.roc,ret = c("accuracy", "spec", "sens", "thr"),x = "local maximas")  Warning message:  In coords.roc(nb2.roc, ret = c("accuracy", "spec", "sens", "thr"), :  The 'transpose' argument to FALSE by default since pROC 1.16. Set transpose = TRUE explicitly to revert to the previous behavior, or transpose = TRUE to silence this warning. Type help(coords\_transpose) for additional information.  > nb2.coords  accuracy specificity sensitivity threshold  1 0.8734177 0.4736842 1.0000000 0.3623130  2 0.8860759 0.6315789 0.9666667 0.4448487  3 0.9367089 0.8947368 0.9500000 0.6560117  4 0.9367089 0.9473684 0.9333333 0.7034914  5 0.8607595 1.0000000 0.8166667 0.7943816  > prob.threshold <- nb2.coords[4,5]  > nb2.pred2 <- ifelse(test = nb2.pred.prob[,1] >= prob.threshold, yes = "No",  + no = "Yes") #... assign the negative class (Yes)  > nb2.pred2 <- as.factor(nb2.pred2)  > nb2.cm2 <- table(actual = test.data$HighSales, predicted = nb2.pred2)  Error in table(actual = test.data$HighSales, predicted = nb2.pred2) :  all arguments must have the same length  > nb2.cm2  Error: object 'nb2.cm2' not found  > ## predicted  > ## actual No Yes  > ## No 50 10  > ## Yes 3 16  > nb2.eval2 <- compute.eval.metrics(nb2.cm2)  Error in compute.eval.metrics(nb2.cm2) : object 'nb2.cm2' not found  > nb2.eval2  Error: object 'nb2.eval2' not found  > data.frame(rbind(nb1.eval, nb2.eval, nb2.eval2),  + row.names = c(paste("NB\_", 1:3, sep = "")))  Error in rbind(nb1.eval, nb2.eval, nb2.eval2) :  object 'nb2.eval2' not found |
|  |
| |  | | --- | | > | |



