Naive-Bayes-Classifier.R

patriciamaya

2020-11-11

```r
#NAIVE BAYES CLASSIFIER
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.0.2
```

```r
attach(Carseats)
head(Carseats)
```

```
##   Sales CompPrice Income Advertising Population Price ShelveLoc Age Education
## 1  9.50       138     73          11        276   120       Bad  42        17
## 2 11.22       111     48          16        260    83      Good  65        10
## 3 10.06       113     35          10        269    80    Medium  59        12
## 4  7.40       117    100           4        466    97    Medium  55        14
## 5  4.15       141     64           3        340   128       Bad  38        13
## 6 10.81       124    113          13        501    72       Bad  78        16
##   Urban  US
## 1   Yes Yes
## 2   Yes Yes
## 3   Yes Yes
## 4   Yes Yes
## 5   Yes  No
## 6    No Yes
```

```r
#create categorical variable for Sales (High, not high)
High<- as.factor(ifelse(Sales>=8, "YES", "NO"))
Carseats <- data.frame(Carseats,High)
Carseats <- Carseats[,-1] #delete first column (Sales col)

#Split train/test
set.seed(2)
indx  <- sample(2,nrow(Carseats), replace=TRUE, prob = c(0.7, 0.3))
train <- Carseats[indx==1, ]
test  <- Carseats[indx==2, ]

#package for naive bayes model
#model 1
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.0.2
```

```r
naive_model <- naiveBayes(High ~ ., data= train)
naive_model
```

```
##
## Naive Bayes Classifier for Discrete Predictors
```

```
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        NO       YES
## 0.5785714 0.4214286
##
## Conditional probabilities:
##      CompPrice
## Y         [,1]     [,2]
##   NO  124.8827 14.58596
##   YES 126.0085 16.21938
##
##      Income
## Y         [,1]     [,2]
##   NO  66.14815 28.16902
##   YES 74.62712 26.52617
##
##      Advertising
## Y         [,1]     [,2]
##   NO  5.425926 5.857315
##   YES 9.186441 7.170630
##
##      Population
## Y         [,1]     [,2]
##   NO  268.3951 150.2537
##   YES 270.3220 145.5607
##
##      Price
## Y         [,1]     [,2]
##   NO  123.9877 21.64163
##   YES 105.3898 22.56615
##
##      ShelveLoc
## Y            Bad       Good     Medium
##   NO  0.38271605 0.06172840 0.55555556
##   YES 0.08474576 0.38135593 0.53389831
##
##      Age
## Y         [,1]     [,2]
##   NO  56.58025 16.15789
##   YES 49.49153 14.88804
##
##      Education
## Y         [,1]     [,2]
##   NO  14.18519 2.551877
##   YES 13.79661 2.636234
##
##      Urban
## Y            No       Yes
##   NO  0.2530864 0.7469136
##   YES 0.3305085 0.6694915
```

```
##
##      US
## Y             No       Yes
##   NO  0.3765432 0.6234568
##   YES 0.2372881 0.7627119
```

```r
#for each variable we get a table of conditional probabilities
#categorical variable--P(Y given X)
#numerical variable-- 1st column average
#                      2nd column StDev
#we use these conditional probabilities for future prediction
pred_class <- predict(naive_model, test, type="class")
pred_class
```

```
##   [1] YES NO  YES YES NO  NO  NO  NO  NO  YES YES YES NO  YES NO  YES YES NO
##  [19] NO  NO  NO  NO  YES NO  YES YES YES NO  NO  NO  NO  NO  NO  NO  NO  NO
##  [37] NO  NO  NO  NO  YES YES NO  NO  NO  NO  YES YES NO  NO  NO  NO  NO  NO
##  [55] NO  NO  NO  NO  NO  YES NO  NO  NO  NO  NO  NO  YES NO  NO  NO  NO  NO
##  [73] YES YES YES NO  NO  NO  YES NO  YES NO  NO  NO  YES NO  YES NO  NO  YES
##  [91] NO  NO  YES YES YES YES YES YES NO  YES YES YES NO  YES NO  NO  YES YES
## [109] YES YES NO  YES YES NO  NO  YES NO  NO  NO  YES
## Levels: NO YES
```

```r
#confusion matrix
table(pred_class, test$High, dnn= c("Prediction", "Actual"))
```

```
##           Actual
## Prediction NO YES
##        NO  63  12
##        YES 11  34
```

```r
#accuracy
(63+34)/(63+34+12+11)  #81% accuracy
```

```
## [1] 0.8083333
```

```r
#predicted probabilities
pred_prob <- predict(naive_model, test, type="raw") #raw
pred_prob
```

```
##                NO        YES
##   [1,] 0.02945217 0.97054783
##   [2,] 0.94509042 0.05490958
##   [3,] 0.46441603 0.53558397
##   [4,] 0.09837505 0.90162495
##   [5,] 0.96126014 0.03873986
##   [6,] 0.91703139 0.08296861
##   [7,] 0.57842696 0.42157304
##   [8,] 0.88984057 0.11015943
##   [9,] 0.78885357 0.21114643
##  [10,] 0.27722189 0.72277811
##  [11,] 0.24341613 0.75658387
##  [12,] 0.27634949 0.72365051
##  [13,] 0.97742066 0.02257934
##  [14,] 0.41206231 0.58793769
##  [15,] 0.93247500 0.06752500
##  [16,] 0.05748086 0.94251914
```

```
## [17,] 0.45153780 0.54846220
## [18,] 0.61236234 0.38763766
## [19,] 0.89700195 0.10299805
## [20,] 0.55551689 0.44448311
## [21,] 0.96346266 0.03653734
## [22,] 0.91367571 0.08632429
## [23,] 0.42509960 0.57490040
## [24,] 0.67000353 0.32999647
## [25,] 0.20405092 0.79594908
## [26,] 0.04592273 0.95407727
## [27,] 0.31807292 0.68192708
## [28,] 0.64265730 0.35734270
## [29,] 0.82741939 0.17258061
## [30,] 0.68021572 0.31978428
## [31,] 0.83660494 0.16339506
## [32,] 0.85331709 0.14668291
## [33,] 0.92262649 0.07737351
## [34,] 0.96049281 0.03950719
## [35,] 0.62671464 0.37328536
## [36,] 0.56163830 0.43836170
## [37,] 0.89309481 0.10690519
## [38,] 0.90505415 0.09494585
## [39,] 0.51778762 0.48221238
## [40,] 0.67384790 0.32615210
## [41,] 0.33451359 0.66548641
## [42,] 0.08183411 0.91816589
## [43,] 0.82783714 0.17216286
## [44,] 0.80612490 0.19387510
## [45,] 0.94677958 0.05322042
## [46,] 0.51347895 0.48652105
## [47,] 0.35620907 0.64379093
## [48,] 0.40570809 0.59429191
## [49,] 0.80261449 0.19738551
## [50,] 0.73324462 0.26675538
## [51,] 0.94061734 0.05938266
## [52,] 0.90061072 0.09938928
## [53,] 0.67321551 0.32678449
## [54,] 0.69031985 0.30968015
## [55,] 0.81005501 0.18994499
## [56,] 0.73507808 0.26492192
## [57,] 0.89027241 0.10972759
## [58,] 0.95845837 0.04154163
## [59,] 0.77893894 0.22106106
## [60,] 0.38792391 0.61207609
## [61,] 0.96348371 0.03651629
## [62,] 0.73103809 0.26896191
## [63,] 0.87743616 0.12256384
## [64,] 0.97748836 0.02251164
## [65,] 0.90041968 0.09958032
## [66,] 0.92046639 0.07953361
## [67,] 0.09293728 0.90706272
## [68,] 0.74733900 0.25266100
## [69,] 0.92144743 0.07855257
## [70,] 0.86837280 0.13162720
```

```
##  [71,] 0.64192519 0.35807481
##  [72,] 0.79583155 0.20416845
##  [73,] 0.18213814 0.81786186
##  [74,] 0.13439811 0.86560189
##  [75,] 0.20256139 0.79743861
##  [76,] 0.64635522 0.35364478
##  [77,] 0.54074808 0.45925192
##  [78,] 0.75432663 0.24567337
##  [79,] 0.09480936 0.90519064
##  [80,] 0.50932696 0.49067304
##  [81,] 0.01622380 0.98377620
##  [82,] 0.56017121 0.43982879
##  [83,] 0.61284924 0.38715076
##  [84,] 0.69821434 0.30178566
##  [85,] 0.17457328 0.82542672
##  [86,] 0.87541517 0.12458483
##  [87,] 0.08390538 0.91609462
##  [88,] 0.52114119 0.47885881
##  [89,] 0.71019509 0.28980491
##  [90,] 0.09747034 0.90252966
##  [91,] 0.59142703 0.40857297
##  [92,] 0.86513969 0.13486031
##  [93,] 0.03532627 0.96467373
##  [94,] 0.11195964 0.88804036
##  [95,] 0.43490019 0.56509981
##  [96,] 0.16372204 0.83627796
##  [97,] 0.09218185 0.90781815
##  [98,] 0.21858348 0.78141652
##  [99,] 0.84569451 0.15430549
## [100,] 0.14364361 0.85635639
## [101,] 0.35453120 0.64546880
## [102,] 0.21121135 0.78878865
## [103,] 0.78006020 0.21993980
## [104,] 0.34690298 0.65309702
## [105,] 0.91488357 0.08511643
## [106,] 0.52142120 0.47857880
## [107,] 0.37412628 0.62587372
## [108,] 0.24265854 0.75734146
## [109,] 0.12729401 0.87270599
## [110,] 0.49396380 0.50603620
## [111,] 0.80453837 0.19546163
## [112,] 0.11248150 0.88751850
## [113,] 0.42343132 0.57656868
## [114,] 0.76081388 0.23918612
## [115,] 0.69664738 0.30335262
## [116,] 0.47656153 0.52343847
## [117,] 0.61453682 0.38546318
## [118,] 0.96210214 0.03789786
## [119,] 0.76112032 0.23887968
## [120,] 0.42245534 0.57754466
```

```r
#model 2 - using laplace estimator
naive_model_laplace <- naiveBayes(High ~ ., data= train, laplace =1)
  #we add 1 instance to each of the categorical variables
```

```r
naive_model_laplace
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        NO       YES
## 0.5785714 0.4214286
##
## Conditional probabilities:
##      CompPrice
## Y         [,1]     [,2]
##   NO  124.8827 14.58596
##   YES 126.0085 16.21938
##
##      Income
## Y         [,1]     [,2]
##   NO  66.14815 28.16902
##   YES 74.62712 26.52617
##
##      Advertising
## Y         [,1]     [,2]
##   NO  5.425926 5.857315
##   YES 9.186441 7.170630
##
##      Population
## Y         [,1]     [,2]
##   NO  268.3951 150.2537
##   YES 270.3220 145.5607
##
##      Price
## Y         [,1]     [,2]
##   NO  123.9877 21.64163
##   YES 105.3898 22.56615
##
##      ShelveLoc
## Y            Bad       Good     Medium
##   NO  0.38181818 0.06666667 0.55151515
##   YES 0.09090909 0.38016529 0.52892562
##
##      Age
## Y         [,1]     [,2]
##   NO  56.58025 16.15789
##   YES 49.49153 14.88804
##
##      Education
## Y         [,1]     [,2]
##   NO  14.18519 2.551877
##   YES 13.79661 2.636234
##
```

```
##      Urban
## Y             No       Yes
##   NO  0.2560976 0.7439024
##   YES 0.3333333 0.6666667
##
##      US
## Y             No       Yes
##   NO  0.3780488 0.6219512
##   YES 0.2416667 0.7583333
```

```
pred_class_laplace <- predict(naive_model_laplace, test, type="class")
pred_class_laplace
```

```
##   [1] YES NO  YES YES NO  NO  NO  NO  NO  YES YES YES NO  YES NO  YES YES NO
##  [19] NO  NO  NO  NO  YES NO  YES YES YES NO  NO  NO  NO  NO  NO  NO  NO  NO
##  [37] NO  NO  NO  NO  YES YES NO  NO  NO  NO  YES YES NO  NO  NO  NO  NO  NO
##  [55] NO  NO  NO  NO  NO  YES NO  NO  NO  NO  NO  NO  YES NO  NO  NO  NO  NO
##  [73] YES YES YES NO  NO  NO  YES NO  YES NO  NO  NO  YES NO  YES NO  NO  YES
##  [91] NO  NO  YES YES YES YES YES YES NO  YES YES YES NO  YES NO  NO  YES YES
## [109] YES NO  NO  YES YES NO  NO  YES NO  NO  NO  YES
## Levels: NO YES
```

```
#confusion matrix
table(pred_class_laplace, test$High, dnn= c("Prediction", "Actual"))
```

```
##           Actual
## Prediction NO YES
##        NO  64  12
##        YES 10  34
```

```
#accuracy
(64+34)/(64+34+12+10)  #81.66% accuracy
```

```
## [1] 0.8166667
```

```
#accuracy only improves much when we have a zero frequency case.

naive_model_laplace$apriori
```

```
## Y
##  NO YES
## 162 118
```

```
#individual conditional probability table
naive_model_laplace$tables$CompPrice
```

```
##      CompPrice
## Y         [,1]     [,2]
##   NO  124.8827 14.58596
##   YES 126.0085 16.21938
```

```
#target variables
naive_model_laplace$levels
```

```
## [1] "NO"  "YES"
```