Portfolio Project 2: Regression File

David Teran & HuyNguyen

Classification

This notebook will use a dataset consisting of 10k or more rows of data and will create classification models based on the dataset.

Created by David Teran & Huy Nguyen on February 15, 2023

Linear models for classifications create decision boundaries to divide the observations into classes. Some strengths to logistic regression is that it can separates classes if they are linearly separable, computationally inexpensive, and nice probabilistic output. The main weakness for logistic regression is it is likely to underfit the data.

First, Read in the dataset to use

```
#install.packages('e1071', dependencies=TRUE)
ClassData <- read.csv("heart_data.csv", na.strings = "NA", header = TRUE)
data(ClassData)
```

```
## Warning in data(ClassData): data set 'ClassData' not found
```

```
#attach(ClassData)
str(ClassData)
```

```
## 'data.frame':    70000 obs. of  14 variables:
##  $ index      : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ id         : int  0 1 2 3 4 8 9 12 13 14 ...
##  $ age        : int  18393 20228 18857 17623 17474 21914 22113 22584 17668 19834 ...
##  $ gender     : int  2 1 1 2 1 1 1 2 1 1 ...
##  $ height     : int  168 156 165 169 156 151 157 178 158 164 ...
##  $ weight     : num  62 85 64 82 56 67 93 95 71 68 ...
##  $ ap_hi      : int  110 140 130 150 100 120 130 130 110 110 ...
##  $ ap_lo      : int  80 90 70 100 60 80 80 90 70 60 ...
##  $ cholesterol: int  1 3 3 1 1 2 3 3 1 1 ...
##  $ gluc       : int  1 1 1 1 1 2 1 3 1 1 ...
##  $ smoke      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ alco       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ active     : int  1 1 0 1 0 0 1 1 1 0 ...
##  $ cardio     : int  0 1 1 1 0 0 0 1 0 0 ...
```

Dividing Data into train/test sets

```
set.seed(1234)
i <- sample(1:nrow(ClassData), nrow(ClassData)*0.80, replace=FALSE)
train <- ClassData[i,]
test <- ClassData[-i,]
```

Using 5 R functions for data exploration Changing some data types to boolean values

```
train$smoke <- as.factor(train$smoke)
train$alco <- as.factor(train$alco)
train$active <- as.factor(train$active)
train$cardio <- as.factor(train$cardio)

test$smoke <- as.factor(test$smoke)
test$alco <- as.factor(test$alco)
test$active <- as.factor(test$active)
test$cardio <- as.factor(test$cardio)

summary(train)
```

```
##      index             id             age            gender
##  Min.   :    0   Min.   :    0   Min.   :10798   Min.   :1.000
##  1st Qu.:17508   1st Qu.:25016   1st Qu.:17665   1st Qu.:1.000
##  Median :35072   Median :50109   Median :19703   Median :1.000
##  Mean   :34996   Mean   :49967   Mean   :19471   Mean   :1.351
##  3rd Qu.:52457   3rd Qu.:74824   3rd Qu.:21329   3rd Qu.:2.000
##  Max.   :69999   Max.   :99999   Max.   :23701   Max.   :2.000
##      height          weight           ap_hi           ap_lo
##  Min.   : 57.0   Min.   : 10.00   Min.   : -120   Min.   :  -70.00
##  1st Qu.:159.0   1st Qu.: 65.00   1st Qu.:  120   1st Qu.:   80.00
##  Median :165.0   Median : 72.00   Median :  120   Median :   80.00
##  Mean   :164.4   Mean   : 74.21   Mean   :  129   Mean   :   96.97
##  3rd Qu.:170.0   3rd Qu.: 82.00   3rd Qu.:  140   3rd Qu.:   90.00
##  Max.   :250.0   Max.   :183.00   Max.   :16020   Max.   :11000.00
##   cholesterol         gluc       smoke     alco      active    cardio
##  Min.   :1.000   Min.   :1.000   0:51075   0:52977   0:11049   0:28045
##  1st Qu.:1.000   1st Qu.:1.000   1: 4925   1: 3023   1:44951   1:27955
##  Median :1.000   Median :1.000
##  Mean   :1.367   Mean   :1.225
##  3rd Qu.:2.000   3rd Qu.:1.000
##  Max.   :3.000   Max.   :3.000
```

```
names(train)
```

```
##  [1] "index"       "id"          "age"         "gender"      "height"
##  [6] "weight"      "ap_hi"       "ap_lo"       "cholesterol" "gluc"
## [11] "smoke"       "alco"        "active"      "cardio"
```

```
cor(train$weight, train$height)
```

```
## [1] 0.2909894
```

```
var(train$weight)
```

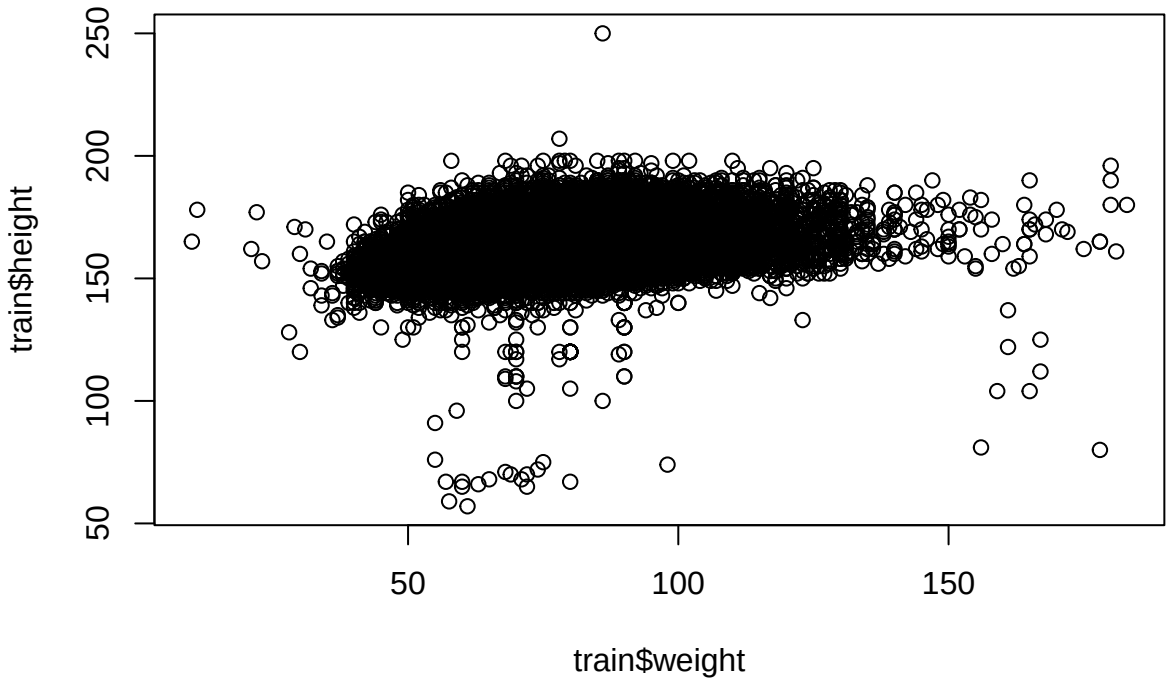
```
## [1] 205.8039
```

```
range(train$weight)
```

```
## [1]  10 183
```

Creating Tables

```
plot(train$weight, train$height)
```




```
hist(train$age)
```

Building the logistic regression model

```
glmHeart <- glm(cardio~., data=train, family="binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glmHeart)
```

```
##
## Call:
## glm(formula = cardio ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -0.9691  -0.1503   0.9993   4.3909
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.184e+00  2.387e-01 -34.284  < 2e-16 ***
## index        2.069e-04  3.911e-04   0.529 0.596736
## id          -1.448e-04  2.739e-04  -0.529 0.597033
## age          1.488e-04  3.952e-06  37.649  < 2e-16 ***
## gender       2.741e-02  2.346e-02   1.169 0.242587
## height      -5.991e-03  1.371e-03  -4.369 1.25e-05 ***
## weight       1.552e-02  7.361e-04  21.086  < 2e-16 ***
```

```
## ap_hi        3.701e-02  6.653e-04  55.627  < 2e-16 ***
## ap_lo        3.283e-04  7.472e-05   4.394 1.11e-05 ***
## cholesterol  5.329e-01  1.669e-02  31.934  < 2e-16 ***
## gluc        -1.261e-01  1.897e-02  -6.647 2.99e-11 ***
## smoke1      -1.388e-01  3.691e-02  -3.761 0.000169 ***
## alco1       -1.618e-01  4.463e-02  -3.625 0.000289 ***
## active1     -2.203e-01  2.339e-02  -9.416  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 77632  on 55999  degrees of freedom
## Residual deviance: 64919  on 55986  degrees of freedom
## AIC: 64947
##
## Number of Fisher Scoring iterations: 8
```

The coefficients quantifies the difference in the log odds of the target variable. The null deviance measures the lack of fit of the model, considering the intercept. Residual deviance is measures the lack of fir for the entire model. Ideally we would want to see the residual deviance much lower than the null deviance. In our model, the residual deviance is lower than the null deviance but there is probably room for improvement.

Building naive Bayes model

```
library(e1071)
nbHeart <- naiveBayes(cardio~., data=train)
nbHeart
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.5008036 0.4991964
##
## Conditional probabilities:
##    index
## Y       [,1]     [,2]
##   0 34941.83 20219.38
##   1 35050.32 20185.23
##
##    id
## Y       [,1]     [,2]
##   0 49890.06 28868.31
##   1 50044.91 28819.76
##
##    age
## Y       [,1]     [,2]
##   0 18887.58 2471.774
```

```
##   1 20055.39 2324.191
##
##    gender
## Y       [,1]      [,2]
##   0 1.346265 0.4757873
##   1 1.355786 0.4787594
##
##    height
## Y       [,1]     [,2]
##   0 164.4801 8.150708
##   1 164.3106 8.254944
##
##    weight
## Y       [,1]     [,2]
##   0 71.61894 13.32536
##   1 76.79947 14.85597
##
##    ap_hi
## Y       [,1]     [,2]
##   0 120.6764 115.4049
##   1 137.4321 198.9715
##
##    ap_lo
## Y        [,1]     [,2]
##   0  84.40799 157.2382
##   1 109.56748 228.4568
##
##    cholesterol
## Y       [,1]      [,2]
##   0 1.216081 0.5260490
##   1 1.518512 0.7764595
##
##    gluc
## Y       [,1]      [,2]
##   0 1.175183 0.5092332
##   1 1.275943 0.6220120
##
##    smoke
## Y            0          1
##   0 0.90768408 0.09231592
##   1 0.91643713 0.08356287
##
##    alco
## Y            0          1
##   0 0.94444643 0.05555357
##   1 0.94759435 0.05240565
##
##    active
## Y           0         1
##   0 0.1822785 0.8177215
##   1 0.2123770 0.7876230
```

A-priori is supposed to show us how likely it is to have cardiovascular disease. Our model shows it as a 50/50 chance to have cardiovascular disease which does not sound correct. The tables below that show

the likelihood data as conditional probabilities. For discrete variables like smoke, alco, and active, it shows the percentages in decimal of how if they have cardiovascular disease or not based on if they smoke, drink alcohol, or are active. The rest are continuous variables so it will show the mean and standard deviation.

Predicting and Evaluating on test data

```
probs <- predict(glmHeart, newdata = test, type = "response")
predHeart <- ifelse(probs>0.5, 1, 0)
acc <- mean(predHeart==test$cardio)
print(paste("accuracy = ", acc))
```

```
## [1] "accuracy =  0.721142857142857"
```

```
table(predHeart, test$cardio)
```

```
##
## predHeart    0    1
##         0 5337 2265
##         1 1639 4759
```

```
predTest <- predict(nbHeart, newdata = test, type = "class")
table(predTest, test$cardio)
```

```
##
## predTest    0    1
##        0 6193 4960
##        1  783 2064
```

```
mean(predTest==test$cardio)
```

```
## [1] 0.5897857
```

Compare the results and indicate why you think these results happened Ideally the predHeart and predTest values would be the same. We did not get this result this is probably because maybe we made the wrong decisions when calculating these values or maybe because the dataset does not fit this model or is too large.

Strengths of Naive Bayes is that it works well in smaller data sets, easy to implement and interpret, and handles high dimensions well. The weaknesses of Naive Bayes is that it is outperformed by other classifiers for larger data sets, predictions are made from the training data, and that the naive assumption that they are independent may not fit the data.

Accuracy is the most common metric to check results. This is the best to tell if a class was miss classified. Kappa tries to adjust accuracy to account for correct predictions by chance. This is used to quantify agreement between two annotators of data. ROC curve is a visualization of performance of the algorithm. This shows the trade off between predicting true positives while also avoiding false positives. The AUC is the area under the curve and this can be us gauge how well it classified the values with 0.5 being the worst and 1 being the best classification.