Homework 2 PSTAT 131

Nicholas Axl Andrian

November 01, 2023

```r
#install.packages("tidyverse")
#install.packages("dplyr")
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(ISLR)
library(ROCR)
library(dplyr)
library(ggplot2)
```

Linear Regression

```r
lm_model <- lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration + year + origin, data
summary(lm_model)
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + year + origin, data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5903 -2.1565 -0.1169  1.8690 13.0604
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -17.218435   4.644294  -3.707  0.00024 ***
## cylinders     -0.493376   0.323282  -1.526  0.12780
## displacement   0.019896   0.007515   2.647  0.00844 **
## horsepower    -0.016951   0.013787  -1.230  0.21963
## weight        -0.006474   0.000652  -9.929  < 2e-16 ***
## acceleration   0.080576   0.098845   0.815  0.41548
## year           0.750773   0.050973  14.729  < 2e-16 ***
## origin         1.426141   0.278136   5.127 4.67e-07 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.328 on 384 degrees of freedom
## Multiple R-squared:  0.8215, Adjusted R-squared:  0.8182
## F-statistic: 252.4 on 7 and 384 DF,  p-value: < 2.2e-16
```

1. Yes, I can reject the null hypothesis as Displacement, Weight, Year and Origin have a smaller p-value than the significance at 0.01, hence allowing us to reject the null hypothesis

```r
pred_val <- predict(lm_model, newdata = Auto)
residuals <- Auto$mpg - pred_val
training_mse <- mean(residuals^2)
training_mse
```

```
## [1] 10.84748
```

2. There is no test MSE as the whole dataset was used as the training set.

```r
str(Auto)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 2
```

```r
head(Auto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

3.

```r
new_data <- data.frame(
  origin = 2,
  cylinders = 4,
  displacement = 132,
  horsepower = 115,
  weight = 3150,
  acceleration = 34,
```

```
  year = 94
)
pred_mpg <- predict(lm_model, newdata = new_data);
pred_mpg
```

```
##        1
## 37.25616
```

4. Using the summary of the coefficients in the lm_model, the difference between the mpg of a japanese vs american car would be 2*1.426141=2.852282, the difference between the mpg of a european and american car would be 1.426141

5. 20*0.019896=0.39792

Algae Classification using Logistic regression

```
algae <- read_table2("algaeBloom.txt", col_names=
c('season','size','speed','mxPH','mnO2','Cl','NO3','NH4',
'oPO4','PO4','Chla','a1','a2','a3','a4','a5','a6','a7'),
na="XXXXXXX")
```

```
## Warning: `read_table2()` was deprecated in readr 2.0.0.
## i Please use `read_table()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   season = col_character(),
##   size = col_character(),
##   speed = col_character(),
##   mxPH = col_double(),
##   mnO2 = col_double(),
##   Cl = col_double(),
##   NO3 = col_double(),
##   NH4 = col_double(),
##   oPO4 = col_double(),
##   PO4 = col_double(),
##   Chla = col_double(),
##   a1 = col_double(),
##   a2 = col_double(),
##   a3 = col_double(),
##   a4 = col_double(),
##   a5 = col_double(),
##   a6 = col_double(),
##   a7 = col_double()
## )
```

```
head(algae)
```

```
## # A tibble: 6 x 18
##   season size  speed  mxPH  mnO2    Cl   NO3   NH4  oPO4   PO4  Chla    a1    a2
##   <chr>  <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 winter small medi~  8      9.8  60.8  6.24 578   105   170    50     0     0
## 2 spring small medi~  8.35   8    57.8  1.29 370   429.  559.    1.3   1.4   7.6
```

```
## 3 autumn small medi~  8.1   11.4  40.0  5.33 347.   126.  187.    15.6   3.3  53.6
## 4 spring small medi~  8.07   4.8  77.4  2.30  98.2  61.2 139.     1.4   3.1  41
## 5 autumn small medi~  8.06   9    55.4 10.4  234.   58.2  97.6  10.5   9.2   2.9
## 6 winter small high   8.25  13.1  65.8  9.25 430    18.2  56.7  28.4  15.1  14.6
## # i 5 more variables: a3 <dbl>, a4 <dbl>, a5 <dbl>, a6 <dbl>, a7 <dbl>
```

```
algae.transformed <- algae %>% mutate_at(vars(4:11), funs(log(.)))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## i Please use a list of either functions or lambdas:
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
algae.transformed <- algae.transformed %>%
mutate_at(vars(4:11),funs(ifelse(is.na(.),median(.,na.rm=TRUE),.)))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## i Please use a list of either functions or lambdas:
##
## # Simple named list: list(mean = mean, median = median)
##
## # Auto named with `tibble::lst()`: tibble::lst(mean, median)
##
## # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# a1 == 0 means low
algae.transformed <- algae.transformed %>% mutate(a1 = factor(as.integer(a1 > 5), levels = c(0, 1)))
```

Starting with the classification task

```
calc_error_rate <- function(predicted.value, true.value){
return(mean(true.value!=predicted.value))
}
```

Train Test Split

```
set.seed(123)
test.indices = sample(1:nrow(algae.transformed), 50)
algae.train=algae.transformed[-test.indices,]
algae.test=algae.transformed[test.indices,]
```

3. logistic regression

```
glm.fit <- glm(a1 ~ . - a2 - a3 - a4 - a5 - a6 - a7, data = algae.train, family = binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = a1 ~ . - a2 - a3 - a4 - a5 - a6 - a7, family = binomial,
##     data = algae.train)
##
```

```
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   4.71508   10.64901   0.443  0.65793
## seasonspring -0.42392    0.88319  -0.480  0.63124
## seasonsummer  0.07914    0.76838   0.103  0.91797
## seasonwinter -0.27821    0.72070  -0.386  0.69948
## sizemedium    0.84100    0.73669   1.142  0.25363
## sizesmall     1.86384    0.83955   2.220  0.02642 *
## speedlow      1.86416    0.85067   2.191  0.02842 *
## speedmedium  -0.42750    0.63722  -0.671  0.50230
## mxPH          0.50548    4.90707   0.103  0.91796
## mnO2          0.01710    0.91780   0.019  0.98513
## Cl           -0.31816    0.38622  -0.824  0.41006
## NO3          -0.14988    0.38149  -0.393  0.69441
## NH4           0.53630    0.30245   1.773  0.07620 .
## oPO4         -1.36654    0.49999  -2.733  0.00627 **
## PO4          -0.51853    0.64300  -0.806  0.42000
## Chla         -0.32295    0.25839  -1.250  0.21134
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 206.98  on 149  degrees of freedom
## Residual deviance: 111.59  on 134  degrees of freedom
## AIC: 143.59
##
## Number of Fisher Scoring iterations: 6
```

Training Data

```
prob.training = predict(glm.fit, type="response")
round(prob.training, digits=2)
```

```
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 0.18 0.05 0.22 0.37 0.46 0.86 0.72 0.70 0.77 0.96 0.98 0.93 0.98 0.98 0.93 0.92
##   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
## 0.87 0.08 0.04 0.99 1.00 0.99 0.99 0.98 0.86 0.99 0.48 0.55 1.00 0.86 1.00 1.00
##   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48
## 0.10 0.09 0.09 0.56 0.02 0.91 0.61 0.92 1.00 0.60 0.91 0.96 0.99 1.00 1.00 1.00
##   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63   64
## 1.00 0.99 1.00 0.98 0.81 1.00 1.00 1.00 1.00 1.00 0.37 0.41 0.26 0.05 0.69 0.45
##   65   66   67   68   69   70   71   72   73   74   75   76   77   78   79   80
## 0.70 0.92 0.47 0.67 0.24 0.12 0.40 0.05 0.08 0.04 0.04 0.07 0.92 0.30 0.45 0.20
##   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96
## 0.62 0.70 0.68 0.80 0.56 0.64 0.36 0.96 0.97 0.99 0.45 0.60 0.91 0.07 0.14 0.05
##   97   98   99  100  101  102  103  104  105  106  107  108  109  110  111  112
## 0.01 0.04 0.55 0.26 0.41 0.10 0.02 0.09 0.66 0.53 0.60 0.62 0.02 0.04 0.85 0.86
##  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128
## 0.23 0.84 0.66 0.56 0.89 0.28 0.57 0.28 0.06 0.33 0.53 0.46 0.13 0.23 0.44 0.61
##  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144
## 0.18 0.66 0.20 0.13 0.40 0.16 0.01 0.04 0.04 0.02 0.01 0.86 0.98 0.94 0.16 0.59
##  145  146  147  148  149  150
## 0.04 0.32 0.43 0.26 0.16 0.03
```

```r
algae.train = algae.train %>%
mutate(pred_a1=as.factor(ifelse(prob.training<=0.5, 0, 1)))
table(pred=algae.train$pred_a1, true=algae.train$a1)
```

```
##     true
## pred  0  1
##    0 55 14
##    1 14 67
```

Testing Data

```r
prob.test <- predict(glm.fit, newdata = algae.test, type = "response")
round(prob.test, digits=2)
```

```
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 0.87 0.97 0.96 0.56 0.29 0.89 0.97 0.03 0.02 0.10 0.88 0.01 0.01 0.52 0.60 0.03
##   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
## 0.26 0.27 0.37 0.97 0.60 0.21 0.51 0.91 0.47 0.71 0.32 0.71 0.95 0.05 0.96 0.08
##   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48
## 0.35 0.97 0.86 0.03 0.88 0.57 0.99 0.19 0.11 0.11 0.98 0.19 0.94 0.02 1.00 0.08
##   49   50
## 0.29 0.58
```

```r
algae.test = algae.test %>%
mutate(pred_a1=as.factor(ifelse(prob.test<=0.5, 0, 1)))
table(pred=algae.test$pred_a1, true=algae.test$a1)
```

```
##     true
## pred  0  1
##    0 17  7
##    1  8 18
```

Error rates

```r
train_error_rate <- calc_error_rate(algae.train$pred_a1, algae.train$a1)
test_error_rate <- calc_error_rate(algae.test$pred_a1, algae.test$a1)
train_error_rate
```

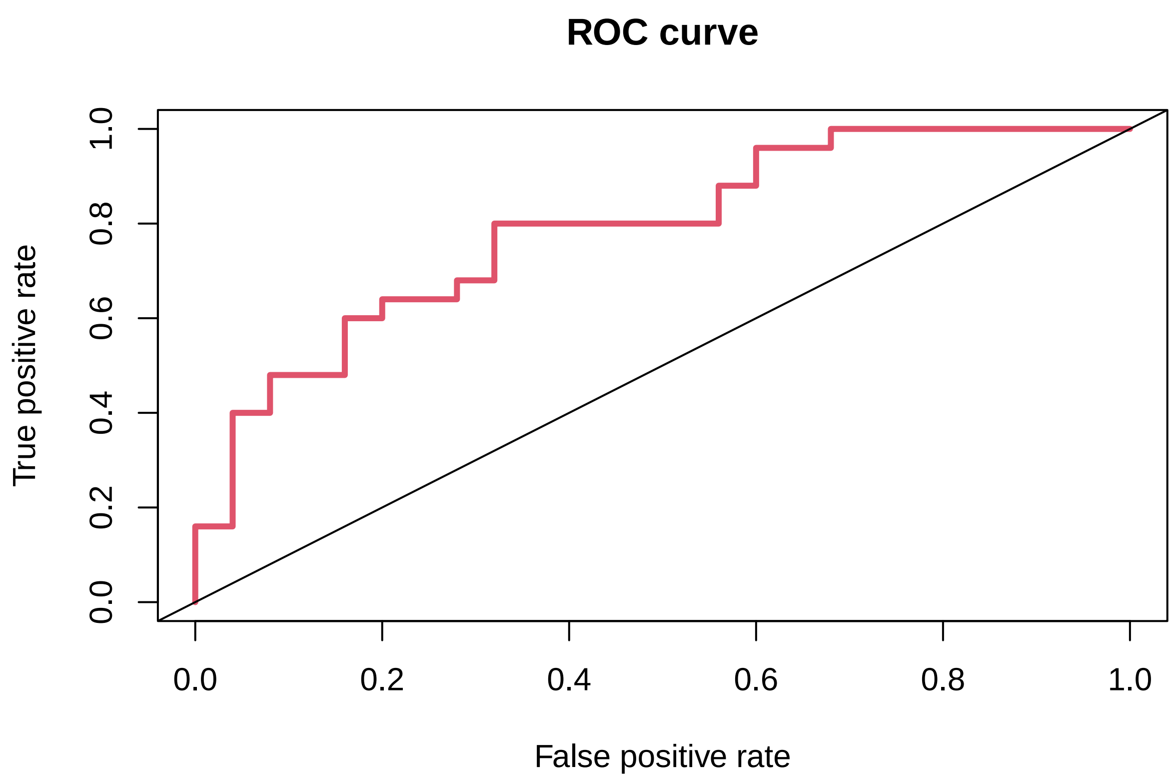
```
## [1] 0.1866667
```

```r
test_error_rate
```

```
## [1] 0.3
```

4. ROC curves

```r
library(ROCR)
pred = prediction(prob.test, algae.test$a1)
perf = performance(pred, measure="tpr", x.measure="fpr")
plot(perf, col=2, lwd=3, main="ROC curve")
abline(0,1)
```



AUC

```
auc = performance(pred, "auc")@y.values
auc
```

```
## [[1]]
## [1] 0.7872
```

Fundamentals of the bootstrap 3.

```
n <- 1000

missing_ratios <- rep(0, 1000)

for (i in 1:1000) {
  bootstrap_sample <- sample(1:n, n, replace = TRUE)
  missing_ratio <- 1 - length(unique(bootstrap_sample)) / n
  missing_ratios[i] <- missing_ratio
}

mean(missing_ratios)
```

```
## [1] 0.367552
```

Cross-validation estimate of test error

1.

```
dat = subset(Smarket, select = -c(Year,Today))
dat$Direction = ifelse(dat$Direction == "Up", 1, 0)
```

```r
set.seed(123)
dat_indice <- sample(1:nrow(dat), 700)
train_dat <- dat[dat_indice, ]
test_dat <- dat[-dat_indice, ]
fit.train <- glm(Direction ~ ., family = binomial, data = train_dat)
summary(fit.train)
```

```
##
## Call:
## glm(formula = Direction ~ ., family = binomial, data = train_dat)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.110793   0.321537  -0.345    0.730
## Lag1        -0.024753   0.064003  -0.387    0.699
## Lag2        -0.047898   0.068797  -0.696    0.486
## Lag3         0.026054   0.068339   0.381    0.703
## Lag4        -0.001578   0.064613  -0.024    0.981
## Lag5        -0.005487   0.065321  -0.084    0.933
## Volume       0.133385   0.211724   0.630    0.529
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 969.12  on 699  degrees of freedom
## Residual deviance: 967.90  on 693  degrees of freedom
## AIC: 981.9
##
## Number of Fisher Scoring iterations: 3
```

```r
prob_test <- predict(fit.train, newdata = test_dat, type = "response")
```

```r
test_dat = test_dat %>%
mutate(pred_testdist=as.factor(ifelse(prob_test<=0.5, 0, 1)))
table(pred=test_dat$pred_testdist, true=test_dat$Direction)
```

```
##     true
## pred   0   1
##    0  42  34
##    1 225 249
```

```r
dat_test_error_rate <- calc_error_rate(test_dat$pred_testdist, test_dat$Direction)
dat_test_error_rate
```

```
## [1] 0.4709091
```

key function

```r
do.chunk <- function(chunkid, folddef, dat, ...){
  # Get training index
  train = (folddef!=chunkid)
  # Get training set and validation set
  dat.train = dat[train, ]
  dat.val = dat[-train, ]
  # Train logistic regression model on training data
  fit.train = glm(Direction ~ ., family = binomial, data = dat.train)
  # get predicted value on the validation set
```

```r
  pred.val = predict(fit.train, newdata = dat.val, type = "response")
  pred.val = ifelse(pred.val > .5, 1,0)
  data.frame(fold = chunkid,
    val.error = mean(pred.val != dat.val$Direction))
}
```

2. Calculating error rate of 10-fold cv

```r
set.seed(123)
nfold = 10;
folds = cut(1:nrow(dat), breaks=nfold, labels=FALSE) %>% sample()
error.folds = NULL;
for (j in seq(10)){
    tmp = do.chunk(chunkid=j, folddef=folds, dat)
    error.folds = rbind(error.folds, tmp) # combine results
}
head(error.folds, 10)
```

```
##    fold val.error
## 1     1 0.4787830
## 2     2 0.4763811
## 3     3 0.4771817
## 4     4 0.4819856
## 5     5 0.4707766
## 6     6 0.4731785
## 7     7 0.4747798
## 8     8 0.4683747
## 9     9 0.4763811
## 10   10 0.4763811
```

```r
mean(error.folds$val.error)
```

```
## [1] 0.4754203
```