# Predictive Modeling

## Series 5

### Exercise 5.1

Show that

$$P(Y|X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}. \tag{1}$$

and, by setting $p(X) = P(Y = 1|X)$,

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X \tag{2}$$

are equivalent.

### Exercise 5.2

This exercise has to do with *odds*.

a) For a randomly chosen person in the U.S. the odds of having blue eyes is 0.47. What fraction of people have blue eyes?

b) 25 % of the U.S. citizens have dark brown eyes. What are the odds to encounter a person with dark brown eyes when randomly chosen?

### Exercise 5.3

Suppose we collect the following data for the students in the course *predictive modeling*

- $X_1$ count the hours studied for the final exam
- $X_2$ is the average mark in the undergraduate studies (”Bachelor”)
- $Y$ is 1 if the final mark in the course is 5.5 or higher and 0 else.

Assume further that we have fit a logistic regression model for $Y$ with the coefficients

$$\hat{\beta}_0 = -6, \quad \hat{\beta}_1 = 0.05, \quad \text{and} \quad \hat{\beta}_2 = 1$$

a) Estimate the probability that a student who studies 40h and has an undergraduate mark of 4.5 has a final examination mark of 5.5 or better.

b) How many hours would the student in (a) need to study to a have a 70% chance of getting a mark of 5.5 or better?

## Exercise 5.4

We consider again the `Default` data set from the lecture notes where the logistic regression model allowing for the predictors `income`, `balance` and `student` is discussed.

a) Generate a downsampled data set with the same random seed as in the lecture notes (i.e. `set.seed(123)` ) in order to compare the results with Example 10.3.4 of the lecture notes.

b) Compute a logistic model that only considers `student` as a predictor variable. Estimate the probability that a student/non-student defaults on his/her debt and compare the result with our findings in the example of the multiple logistic regression model `default~balance+income+student` (cf. Example 10.5.1).

c) Generate two boxplots with respect to `balance` for students and non-students and explain the counterintuitive result above.

## Exercise 5.5

We will now develop a model to predict whether a given car gets high or low gas mileage based on the `Auto` data set.

a) Get familiar with the Auto data set by typing `library(ÏSLR")`, then `?Auto` in the `R`-console.

b) Create a binary variable, `mpg01`, that contains 1 if `mpg` contains a value above its median and 0 else. Create a new data frame that contains `mpg01` and the other `Auto` variables except `mpg` and `name`.

c) Explore the data graphically to get a first idea on which variabels are correlated with `mpg01`. Try scatter- and parallel coordinate plots. The latter can be done by the `parcoord` command in the `MASS` package.

d) Split the data into training and test set.

e) Compute a logistic regression model with the training set. Compute the classification error on the training and the test set and compare it with the cross-validated error.

## Exercise 5.6

In this exercise we study an important tool to assess the performance of a binary classifier and to chose the optimal threshold for the class probabilities: the *receiver operating characteristic (ROC)*-curve. We will use the `Auto` data from the previous exercise, with `mpg` replaced by `mpg01`.

a) Use the same training and test set as in the previous exercise and compute the confusion matrix for both sets.

b) Determine the classification accuracy, recall, sensitivity and F1 score on the basis of the test confusion matrix. Consider a 1 as positive result.

c) Write an `R` -function that computes for a given $\alpha \in [0, 1]$ and class probabilities the classification result with threshold $\alpha$.

d) Now loop a threshold value $\alpha$ from 0 to 1 and compute the classification results and store for each $\alpha$ the true positive rate (*recall*) and the false positive rate. Plot the true positive rate against the false positive rate. The resulting curve is called receiver operating characteristic (ROC) curve.
**Hint:** The *true positive rate* is defined as:

$$\frac{\text{TP}}{\text{TP+FN}}$$

The *false positive rate* is defined as:

$$\frac{\text{FP}}{\text{FP+TN}}$$

e) Interpret the ROC-curve: How can it be used in order to assess the quality of a binary classifier? Give a rule on choosing the optimal threshold parameter.

# Result Checker

**E 5.2**:

a) $p \approx 0.32$

b) $o \approx 0.33$

**E 5.3**:

a) $p = 0.622$

b) $\approx 47$ h

**E 5.4**:

b) Student: 54.27% , non-Student: 47.69%

**E 5.5**:

e) Approximate Values:

| | training | testing | cross-validation |
|---|---|---|---|
| error | 7.96% | 11.54% | 9.3% |

# Predictive Modeling

## Solutions to Series 5

**Solution 5.1** Setting $p(X) = P(Y = 1|X)$ we find from (1) by multiplying both sides with $1 + e^{\beta_0+\beta_1 X}$ that

$$p(X)(1 + e^{\beta_0+\beta_1 X}) = e^{\beta_0+\beta_1 X}.$$

Subtracting $p(X)e^{\beta_0+\beta_1 X}$ then yields

$$p(X) = e^{\beta_0+\beta_1 X} - p(X)e^{\beta_0+\beta_1 X} = e^{\beta_0+\beta_1 X}(1 - p(X)).$$

Dividing by $1 - p(X)$ and taking the log on both sides eventually gives

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X.$$

**Solution 5.2** The relation of the odds $o$ and the proportion $p$ has already been deduced in Exercise 1:

$$p = \frac{o}{1+o}, \quad \Leftrightarrow \quad o = \frac{p}{1-p}.$$

a) $p = 0.47/(1 + 0.47) \approx 0.32$.

b) $o = 0.25/(1 - 0.25) \approx 0.33$.

**Solution 5.3**

a) We estimate the probability by the logistic formula

$$p(X_1, X_2) = \frac{e^{\hat\beta_0+\hat\beta_1 X_1+\hat\beta_2 X_2}}{1 + e^{\hat\beta_0+\hat\beta_1 X_1+\hat\beta_2 X_2}}$$

This gives $p(40, 4.5) = 0.622$.

b) From the log-odds formula we find

$$\log\left(\frac{p(X_1, X_2)}{1 - p(X_1, X_2)}\right) = \hat\beta_0 + \hat\beta_1 X_1 + \hat\beta_2 X_2.$$

For $p(X_1, X_2) = 0.7$ and $X_2 = 4.5$ this gives

$$X_1 = \frac{1}{\hat\beta_1}\left(\log\left(\frac{0.7}{0.3}\right) - \hat\beta_0 - \hat\beta_2 4.5\right) \approx 47\text{h}.$$

## Solution 5.4

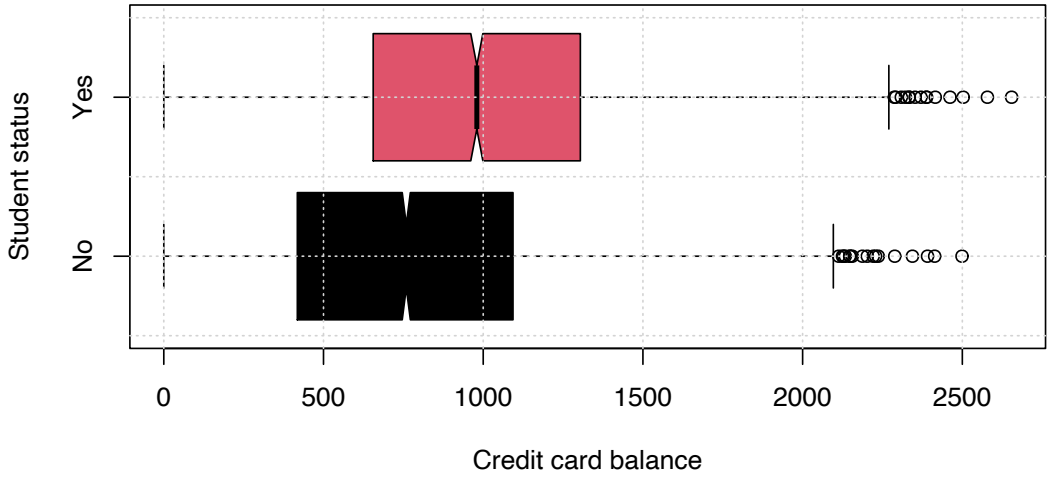
a)
```
require(ISLR)
set.seed(123)

# resample
idx.yes = which(Default$default == "Yes")
idx.no = sample(which(Default$default == "No"), replace = F,
    size = 333)
idx = c(idx.yes, idx.no)

Default.ds = Default[idx, ]
```

b)
```
glm.fit.stud <- glm(default ~ student, family = binomial,
    data = Default.ds)

# coefficients
coeffs = coefficients(glm.fit.stud)
coeffs

## (Intercept)  studentYes
##  -0.1527628   0.4643503

# probabilites
prob.stud = 1/(1 + exp(-(coeffs[1] + coeffs[2] * 1)))
prob.non.stud = 1/(1 + exp(-(coeffs[1] + coeffs[2] * 0)))
prob.stud

## (Intercept)
##   0.5772727

prob.non.stud

## (Intercept)
##   0.4618834
```

It turns out that students are more likely (57.73%) to default on their debt than non-students (46.19%) which can already be seen from the positive coefficient for **student**. This somehow seems to contradict our findings in the multiple logistic regression model from the lecture notes, where the coefficient for **student** was negative.

c)
```
boxplot(balance ~ student, data = Default, col = Default$student,
    notch = TRUE, horizontal = TRUE, ylab = "Student status",
    xlab = "Credit card balance")
```

```
grid()
```




The boxplot nicely indicates that the credit card balance is correlated with the student status. Students tend to hold higher levels of debts which is in turn associated with a higher probability of default. This means that an individual student with a given credit card balance will tend to have lower probability to default than a non-student with the same balance. Students *on the whole* hold a higher risk to default because on average they have higher credit card balances, an *individual* student, however, is more trustworthy than a non-student with the same credit card balance.

## Solution 5.5

a)

b)

```
print(names(Auto))
```

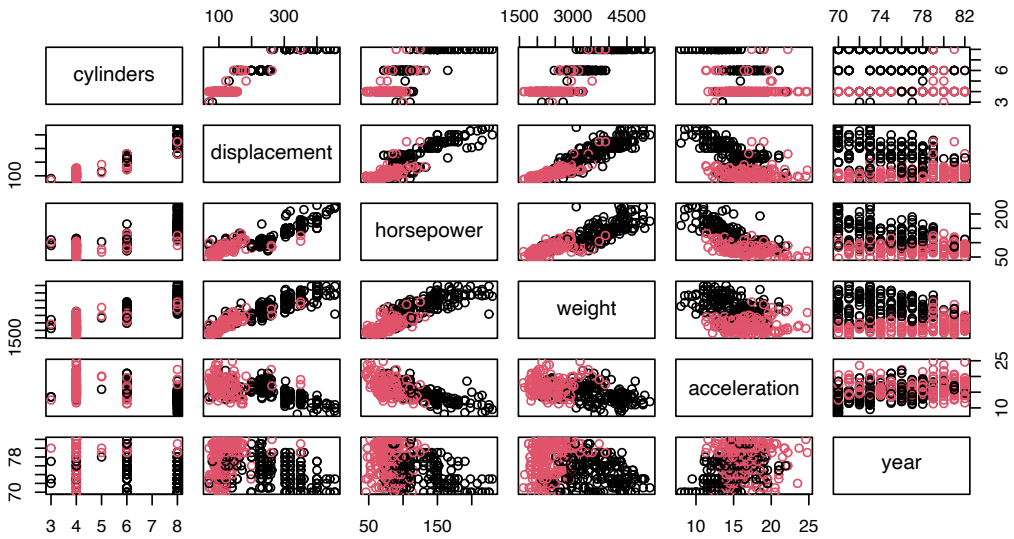
```
## [1] "mpg"          "cylinders"    "displacement"
## [4] "horsepower"   "weight"       "acceleration"
## [7] "year"         "origin"       "name"
```

```
# new data frame
Auto1 <- Auto[, !(names(Auto) == "mpg" | names(Auto) ==
    "name")]
Auto1$mpg01 <- as.numeric(Auto$mpg > median(Auto$mpg))
head(Auto1)
```

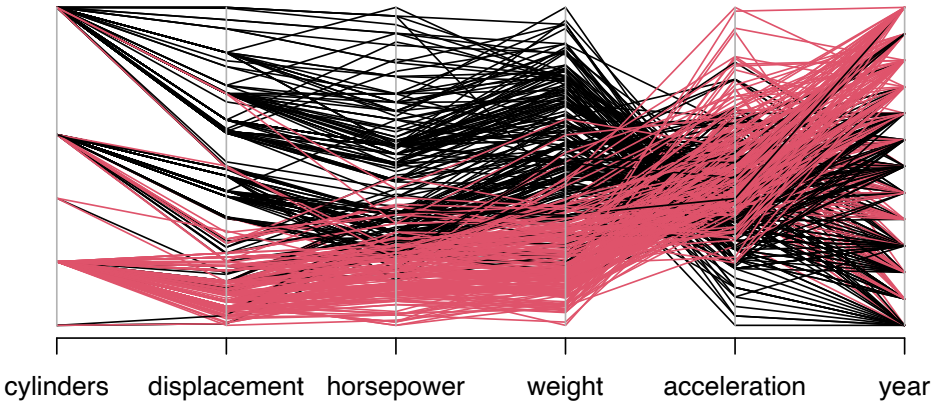
```
##   cylinders displacement horsepower weight
## 1         8          307        130   3504
## 2         8          350        165   3693
## 3         8          318        150   3436
## 4         8          304        150   3433
## 5         8          302        140   3449
## 6         8          429        198   4341
##   acceleration year origin mpg01
## 1         12.0   70      1     0
## 2         11.5   70      1     0
## 3         11.0   70      1     0
## 4         12.0   70      1     0
## 5         10.5   70      1     0
## 6         10.0   70      1     0
```

c) We start by examining the numeric predictors

```
# leave out non-numeric variables
idx = (names(Auto1) == "mpg01") | (names(Auto1) == "origin")

# scatterplot
plot(Auto1[, !idx], col = Auto1$mpg01 + 1)
```




```
# parallel coordinates
require(MASS)
parcoord(Auto1[, !idx], col = Auto1$mpg01 + 1)
```




According to the scatter and parallel coordinate plots, we observe that the two classes are separable (according to the concentration of data points with the same color). These plots indicate that there might be good predictors for **mpg01** in the data set. The correlation between the fuel consumption and the origin of the car can be studied by a contingency table

```
orig = c("American", "European", "Japanese")
addmargins(table(as.integer(Auto1$mpg01), orig[Auto1$origin],
    dnn = c("mpg01", "origin")))
```

```
##      origin
## mpg01 American European Japanese Sum
##    0        173       14        9 196
##    1         72       54       70 196
##    Sum      245       68       79 392
```

The table tells us, that Japanese and European cars in the data set have a high mileage compared to the american cars. In particular, **mpg01** and **origin** are highly correlated.

d) We randomly split the data in a training and a test set. We choose a 4 : 1 ratio

```
idx.test = sample(1:length(Auto[, 1]), size = round(length(Auto[,
    1])/5), replace = F)

testing = Auto1[idx.test, ]
training = Auto1[-idx.test, ]
```

e) We first compute the model and examine the coefficients (for the sake of lucidity we only print out the estimates and the $p$-values)

```
glm.fit = glm(mpg01 ~ ., data = training, family = binomial)
s = summary(glm.fit)
s$coefficients[, c(1, 4)]

##                    Estimate     Pr(>|z|)
## (Intercept)  -15.174361046 2.004378e-02
## cylinders     -0.561229539 2.227589e-01
## displacement   0.007891501 5.445222e-01
## horsepower    -0.043033873 1.028207e-01
## weight        -0.004167857 1.327016e-03
## acceleration  -0.009971534 9.529616e-01
## year           0.420761619 7.048931e-07
## origin         0.380913627 3.512376e-01
```

We see that **weight** and **year** are significant factors in the model. We next compute the classification errors

```
# classification error
cost <- function(true.class, est.prob) {
    mean((est.prob < 0.5) & true.class == 1 | (est.prob >
        0.5) & true.class == 0)
}

# training error
pred.prob = predict(glm.fit, type = "response")
true.class = as.integer(training$mpg01)
error.train = cost(true.class, pred.prob)

# test error
pred.prob = predict(glm.fit, newdata = testing, type = "response")
true.class = as.integer(testing$mpg01)
error.test = cost(true.class, pred.prob)

# cv error
require(boot)
set.seed(123)
error.cv <- cv.glm(glmfit = glm.fit, data = training, cost = cost,
    K = 5)
```

The errors can be summarized as follows

| | training | testing | cross-validation |
|---|---|---|---|
| error | 9.55% | 10.26% | 10% |

## Solution 5.6

a)
```
# training error
pred.class <- as.integer(predict(glm.fit, type = "response") >
    0.5)
true.class <- as.integer(training$mpg01)
addmargins(table(pred.class, true.class))

##            true.class
## pred.class   0   1 Sum
##        0   139  11 150
##        1    19 145 164
##        Sum 158 156 314

# test error
pred.class <- as.integer(predict(glm.fit, newdata = testing,
    type = "response") > 0.5)
true.class <- as.integer(testing$mpg01)
cm_test <- addmargins(table(pred.class, true.class))
```

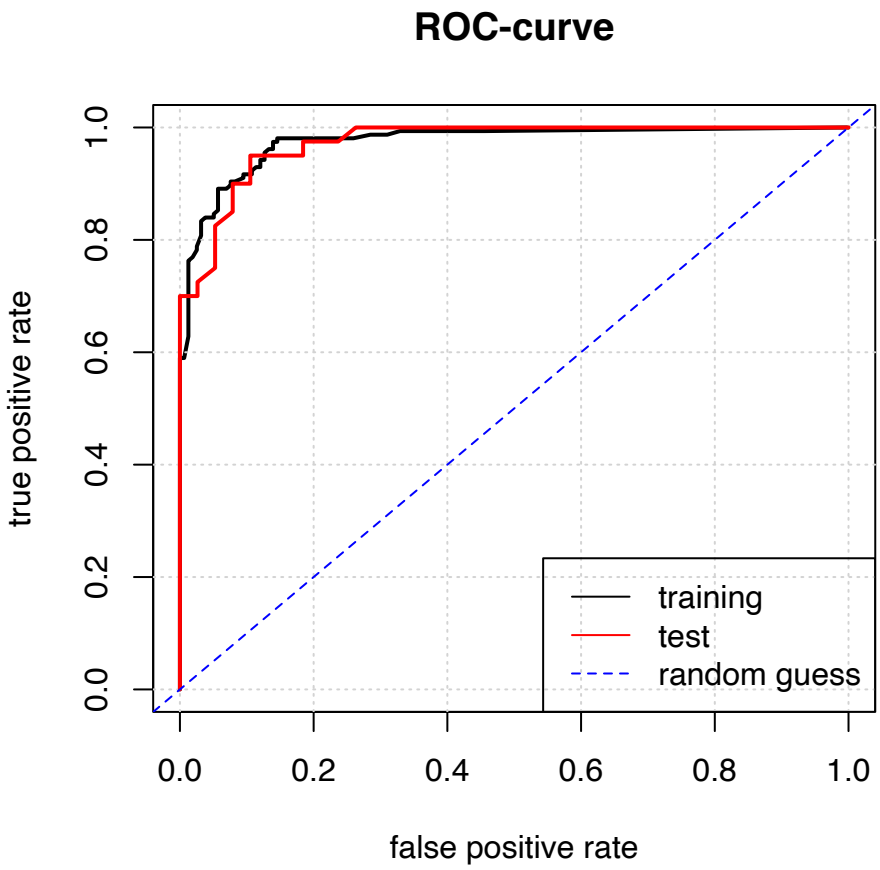
b)
```
# Accuracy : (tp + tn) (tp + fp + fn + tn )
accuracy <- ((cm_test[2, 2] + cm_test[1, 1])/(cm_test[2,
    2] + cm_test[1, 1] + cm_test[2, 1] + cm_test[1, 2]))
cat("Accuracy : ", accuracy, "\n")

## Accuracy :  0.8974359

# Precision: tp/(tp+fp):
precision <- cm_test[1, 1]/sum(cm_test[1, 1:2])
cat("Precision : ", precision, "\n")

## Precision :  0.8947368

# Recall: tp/(tp + fn):
recall <- cm_test[1, 1]/sum(cm_test[1:2, 1])
cat("Recall : ", recall, "\n")

## Recall :  0.8947368

# F1-Score: 2 * precision * recall /(precision +
# recall):
f1_score <- 2 * precision * recall/(precision + recall)
cat("F1 score : ", f1_score, "\n")
```

```
## F1 score :  0.8947368
```

c)
```
class = function(alpha, prob) {
    return(as.integer(prob > alpha))
}
```

d)
```
# threshold values
alpha = seq(0, 1, by = 0.01)

# ROC curve for the training set
pred.prob = predict(glm.fit, type = "response")
true.class = as.integer(training$mpg01)
# number of cars with mpg > median
n.true = sum(true.class)
# number of cars with mpg < median
n.false = sum(1 - true.class)
tpr = vector()
fpr = vector()

for (t in alpha) {
    pred.class = class(t, pred.prob)
    # true positive rate = recall (sensitivity) =
    # among all cars with mpg > median, how many were
    # predicted to have mpg > median
    cur.tpr = sum((pred.class == 1) & (true.class == 1))/n.true
    # false positive rate = among all cars with mpg <
    # median how many were incorrectly predicted to
    # have mpg > median
    cur.fpr = sum((pred.class == 1) & (true.class == 0))/n.false

    tpr = c(tpr, cur.tpr)
    fpr = c(fpr, cur.fpr)
}
plot(fpr, tpr, type = "l", lwd = 2, main = "ROC-curve",
    xlab = "false positive rate", ylab = "true positive rate")
grid()

# ROC curve for the test set
pred.prob = predict(glm.fit, newdata = testing, type = "response")
true.class = as.integer(testing$mpg01)
n.true = sum(true.class)
n.false = sum(1 - true.class)
```

```
tpr.test = vector()
fpr.test = vector()

for (t in alpha) {
    pred.class = class(t, pred.prob)
    # true positive rate = recall (sensitivity) =
    # among all cars with mpg > median, how many were
    # predicted to have mpg > median
    cur.tpr = sum((pred.class == 1) & (true.class == 1))/n.true
    # false positive rate = among all cars with mpg <
    # median how many were incorrectly predicted to
    # have mpg > median
    cur.fpr = sum((pred.class == 1) & (true.class == 0))/n.false

    tpr.test = c(tpr.test, cur.tpr)
    fpr.test = c(fpr.test, cur.fpr)
}
lines(fpr.test, tpr.test, lwd = 2, col = "red")


# draw random classification line
abline(a = 0, b = 1, lty = 2, col = "blue")

# add legend
legend(x = "bottomright", legend = c("training", "test",
    "random guess"), col = c("black", "red", "blue"), lty = c(1,
    1, 2))
```




e) Each binary classifier corresponds to a point in the ROC-space (the unit square). The left upper corner of the ROC-space contains the best classifiers for they have a true positive rate of 100% and a false positive rate of 0. The diagonal line corresponds to random classifiers. So for the classification method *on the whole* it is best to have a ROC-curve that runs as close as possible to the left and upper edge of the square. The area under curve (AUC) is often used to measure the quality of the method. The closest point on the curve to $(0, 1)$ then corresponds to the best threshold and in turn to the best classifier.

```
# AUC
n = length(tpr)
AUC = sum(-tpr[1:n - 1] * diff(fpr))
AUC

## [1] 0.9765093

n.test = length(tpr.test)
AUC.test = sum(-tpr.test[1:n - 1] * diff(fpr.test))
```

```
AUC.test
```

```
## [1] 0.9736842
```

```
# best classifier (threshold)
dist = fpr^2 + (1 - tpr)^2
alpha.best = alpha[which.min(dist)]
alpha.best
```

```
## [1] 0.6
```

```
dist.test = fpr.test^2 + (1 - tpr.test)^2
alpha.best.test = alpha[which.min(dist.test)]
alpha.best.test
```

```
## [1] 0.32
```