Abhishek HW4

```r
# This question should be answered using the Default data set. In Chapter 4
on classification, we used logistic regression to predict the probability of
default using income and balance. Now we will estimate the test error of this
logistic regression model using the validation set approach. Do not forget to
set a random seed before beginning your analysis.

# (a) Fit a logistic regression model that predicts default using income and
balance.

library(ISLR)
attach(Default)
head(Default)

##   default student   balance    income
## 1      No      No  729.5265 44361.625
## 2      No     Yes  817.1804 12106.135
## 3      No      No 1073.5492 31767.139
## 4      No      No  529.2506 35704.494
## 5      No      No  785.6559 38463.496
## 6      No     Yes  919.5885  7491.559

set.seed(1)
log_fit <- glm(default ~ income + balance, data = Default, family =
"binomial")
summary(log_fit)

##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 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: 2920.6  on 9999  degrees of freedom
```

```
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8

# (b) Using the validation set approach, estimate the test error of this
model. You need to perform the following steps:

# i. Split the sample set into a training set and a validation set.

indices <- sample(nrow(Default),nrow(Default)*0.5)
train <- Default[indices,]
test <- Default[-indices,]

# ii. Fit a logistic regression model using only the training data set.

log_fit <- glm(default ~ income + balance,family = binomial,data=train)

# iii. Obtain a prediction of default status for each individual in the
validation set using a threshold of 0.5.

log_proba <- predict(log_fit, test, type="response")
log_predict <- ifelse(log_proba > 0.5,"Yes","No")


# iv. Compute the validation set error, which is the fraction of the
observations in the validation set that are misclassified.

table(test$default, log_predict, dnn=c("Actual","Predicted"))

##       Predicted
## Actual   No  Yes
##    No  4805   28
##    Yes  115   52

mean(log_predict != test$default)

## [1] 0.0286

# (c) Repeat the process in (b) three times, using three different splits of
the observations into a training set and a validation set. Comment on the
results obtained.

indices <- sample(nrow(Default),nrow(Default)*0.9)
train <- Default[indices,]
test <- Default[-indices,]
log_fit <- glm(default ~ income + balance,family = binomial,data=train)
log_proba <- predict(log_fit, test, type="response")
log_predict <- ifelse(log_proba > 0.5,"Yes","No")
table(test$default, log_predict, dnn=c("Actual","Predicted"))
```

```
##       Predicted
## Actual  No Yes
##    No  958   9
##    Yes  22  11
```

```r
mean(log_predict != test$default)
```

```
## [1] 0.031
```

```r
indices <- sample(nrow(Default),nrow(Default)*0.7)
train <- Default[indices,]
test <- Default[-indices,]
log_fit <- glm(default ~ income + balance,family = binomial,data=train)
log_proba <- predict(log_fit, test, type="response")
log_predict <- ifelse(log_proba > 0.5,"Yes","No")
table(test$default, log_predict, dnn=c("Actual","Predicted"))
```

```
##       Predicted
## Actual   No  Yes
##    No  2889   16
##    Yes   69   26
```

```r
mean(log_predict != test$default)
```

```
## [1] 0.02833333
```

```r
indices <- sample(nrow(Default),nrow(Default)*0.4)
train <- Default[indices,]
test <- Default[-indices,]
log_fit <- glm(default ~ income + balance,family = binomial,data=train)
log_proba <- predict(log_fit, test, type="response")
log_predict <- ifelse(log_proba > 0.5,"Yes","No")
table(test$default, log_predict, dnn=c("Actual","Predicted"))
```

```
##       Predicted
## Actual   No  Yes
##    No  5775   16
##    Yes  142   67
```

```r
mean(log_predict != test$default)
```

```
## [1] 0.02633333
```

```r
# We see that the test error rate's are variable.

# (d) Consider another logistic regression model that predicts default using
income, balance and student (qualitative). Estimate the test error for this
model using the validation set approach. Does including the qualitative
variable student lead to a reduction of test error rate?

indices <- sample(nrow(Default), nrow(Default)*0.5)
train <- Default[indices,]
```

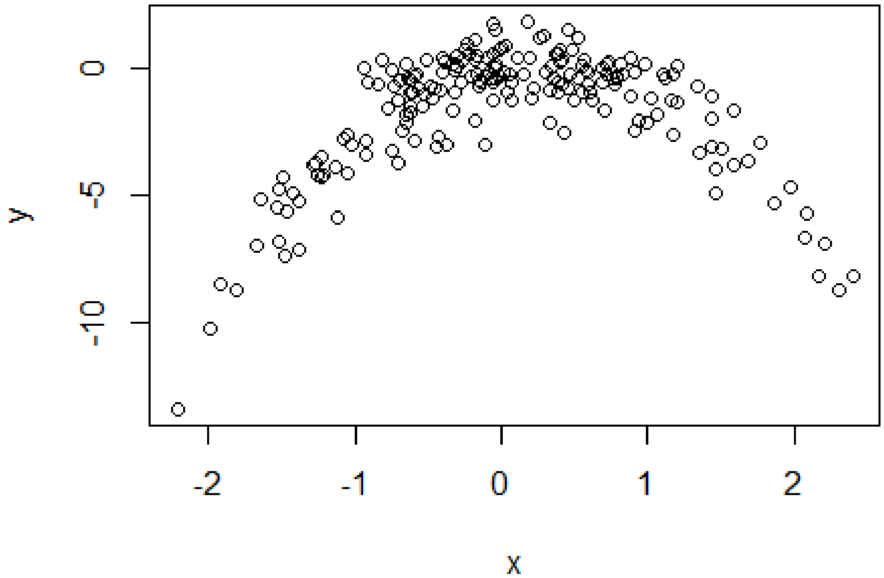
```r
test <- Default[-indices,]
log_fit <- glm(default ~ income + balance + student, family = binomial,
data=train)
log_proba <- predict(log_fit, test, type="response")
log_predict <- ifelse(log_proba > 0.5,"Yes","No")
table(test$default, log_predict, dnn=c("Actual","Predicted"))
```

```
##       Predicted
## Actual   No  Yes
##    No  4829   22
##    Yes  101   48
```

```r
mean(log_predict != test$default)
```

```
## [1] 0.0246
```

```r
# The addition of the “student” dummy variable doesn't lead to a reduction in
the test error rate.

# This question requires performing cross validation on a simulated data set.
# (a) Generate a simulated data set as follows:
#      set.seed(1)
#      x=rnorm(200)
#      y=x-2*x^2+rnorm(200)
# In this data set, what is ▯ and what is ▯? Write out the model used to
generate the data in equation form (i.e., the true model of the data).

set.seed(1)
x = rnorm(200)
y = x-2*x^2+rnorm(200)

# n=200 and p=2, the model used is Y = X – 2X2 + εrror.

# (b) Create a scatter plot of ▯ vs ▯. Comment on what you find.

plot(x, y)
```




```
# There is a curved(non-linear) relationship.

# (c) Consider the following four models for the data set:
#         i. � = �0 + �1� + �
#        ii. � = �0 + �1� + �2�2 + �
#       iii. � = �0 + �1� + �2�2 + �3�3 + �
#        iv. � = �0 + �1� + �2�2 + �3�3 + �4�4 + �
#  Compute the LOOCV errors that result from fitting these models.

library(boot)
set.seed(1)
data <- data.frame(x, y)
cv_error = rep(0,4)
for (i in 1:4){
glm_fit = glm(y~poly(x,i), data=data)
cv_error[i] = cv.glm(data, glm_fit)$delta[1]
}
cv_error

## [1] 6.037638 1.040922 1.039049 1.028604

# (d) Repeat (c) using another random seed, and report your results. Are your
results the same as what you got in (c)? Why?

library(boot)
set.seed(20)
data <- data.frame(x, y)
```

```r
cv_error = rep(0,4)
for (i in 1:4){
glm_fit = glm(y~poly(x,i), data=data)
cv_error[i] = cv.glm(data, glm_fit)$delta[1]
}
cv_error

## [1] 6.037638 1.040922 1.039049 1.028604

# The results above are identical to the results obtained in (c). While
performing LOOCV multiple times will always yield the same results, because
we split based on one observation each time.

# (e) Which of the models in (c) has the smallest LOOCV error? Is this what
you expected? Explain your answer.

# The LOOCV error is minimum for the 2nd order polynomial function. This is
expected since we saw in (b) that the relation between "x" and "y" is
quadratic.

# (f) Now we use 5-fold CV for the model selection. Compute the CV errors
that result from fitting the four models. Which model has the smallest CV
error? Are the results consistent with LOOCV?

library(boot)
set.seed(1)
kcv_error = rep(0,4)
data <- data.frame(x, y)
for (i in 1:4) {
glm_fit = glm(y~poly(x,i), data=data)
kcv_error[i] = cv.glm(data, glm_fit, K=5)$delta[1]
}
kcv_error

## [1] 5.888437 1.036580 1.039166 1.015776

# The 5-fold cross validation error is minimum for the 2nd order polynomial
function. Also, the results are in consistent with LOOCV.

# (g) Repeat (f) using 10-fold CV. Are the results the same as 5-fold CV?

library(boot)
set.seed(1)
kcv_error = rep(0,4)
data <- data.frame(x, y)
for (i in 1:4) {
glm_fit = glm(y~poly(x,i), data=data)
kcv_error[i] = cv.glm(data, glm_fit, K=10)$delta[1]
}
kcv_error
```

```
## [1] 5.968520 1.043498 1.035259 1.018977

# There is not much of a difference compared to error obtained in (f).
```