# HW4 - ISEN 613

*Gourav Ghoshal*

*October 17, 2018*

## Problem 1

**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.

```r
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.4.2
```

```r
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
```

```r
attach(Default)

# model fitting on whole data
lr_model0 = glm(default~balance+income, data = Default, family = binomial)
summary(lr_model0)
```

```
##
## Call:
## glm(formula = default ~ balance + income, 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 ***
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## ---
## 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.

```r
# spliting
set.seed(42)
n = dim(Default)[1]
indx = sample.int(n, floor(0.20*n))
train1 = Default[-indx,]
dim(train1)
```

```
## [1] 8000    4
```

```r
test1 = Default[indx,]
dim(test1)
```

```
## [1] 2000    4
```

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

```r
# model fitting
lr_model1 = glm(default ~ balance+income, data = train1, family = binomial)
print(contrasts(default))
```

```
##     Yes
## No    0
## Yes   1
```

```r
summary(lr_model1)
```

```
##
## Call:
## glm(formula = default ~ balance + income, family = binomial,
##     data = train1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4719  -0.1416  -0.0574  -0.0215   3.7235
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.161e+01  4.859e-01 -23.895  < 2e-16 ***
## balance      5.609e-03  2.525e-04  22.218  < 2e-16 ***
```

```
## income       2.400e-05  5.654e-06   4.244 2.19e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2300.0  on 7999  degrees of freedom
## Residual deviance: 1240.3  on 7997  degrees of freedom
## AIC: 1246.3
##
## Number of Fisher Scoring iterations: 8
```

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

```
pred_lr1 = predict(lr_model1,test1, type = 'response')
pred_class1 = ifelse(pred_lr1<0.5, 'No', 'Yes')
```

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

```
mean(test1$default != pred_class1)
```

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

### (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

```
# 90 - 10 split
## spliting
n = dim(Default)[1]
indx = sample.int(n, floor(0.10*n))
train2 = Default[-indx,]
dim(train2)
```

```
## [1] 9000    4
```

```
test2 = Default[indx,]
dim(test2)
```

```
## [1] 1000    4
```

```
## model fitting
lr_model2 = glm(default ~ balance+income, data = train2, family = binomial)
print(contrasts(default))
```

```
##     Yes
## No    0
## Yes   1
```

```r
summary(lr_model2)
```

```
##
## Call:
## glm(formula = default ~ balance + income, family = binomial,
##     data = train2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4474  -0.1460  -0.0590  -0.0219   3.7087
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.144e+01  4.544e-01 -25.183  < 2e-16 ***
## balance      5.569e-03  2.365e-04  23.546  < 2e-16 ***
## income       2.129e-05  5.262e-06   4.046  5.2e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2623.9  on 8999  degrees of freedom
## Residual deviance: 1429.5  on 8997  degrees of freedom
## AIC: 1435.5
##
## Number of Fisher Scoring iterations: 8
```

```r
## error checking
pred_lr2 = predict(lr_model2,test2, type = 'response')
pred_class2 = ifelse(pred_lr2<0.5, 'No', 'Yes')
mean(test2$default != pred_class2)
```

```
## [1] 0.023
```

```r
####################################################

# 75 - 25 split
n = dim(Default)[1]
indx = sample.int(n, floor(0.25*n))
train3 = Default[-indx,]
dim(train3)
```

```
## [1] 7500    4
```

```r
test3 = Default[indx,]
dim(test3)
```

```
## [1] 2500    4
```

```r
## model fitting
lr_model3 = glm(default ~ balance+income, data = train3, family = binomial)
print(contrasts(default))
```

```
##     Yes
## No    0
## Yes   1
```

```r
summary(lr_model3)
```

```
##
## Call:
## glm(formula = default ~ balance + income, family = binomial,
##     data = train3)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4001  -0.1380  -0.0548  -0.0202   3.7493
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.153e+01  5.120e-01  -22.53  < 2e-16 ***
## balance      5.620e-03  2.669e-04   21.06  < 2e-16 ***
## income       1.828e-05  5.861e-06    3.12  0.00181 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2083.3  on 7499  degrees of freedom
## Residual deviance: 1121.8  on 7497  degrees of freedom
## AIC: 1127.8
##
## Number of Fisher Scoring iterations: 8
```

```r
## error checking
pred_lr3 = predict(lr_model3,test3, type = 'response')
pred_class3 = ifelse(pred_lr3<0.5, 'No', 'Yes')
mean(test3$default != pred_class3)
```

```
## [1] 0.0316
```

```r
#####################################################
# 50 - 50 split
set.seed(42)
n = dim(Default)[1]
indx = sample.int(n, floor(0.50*n))
train4 = Default[-indx,]
dim(train4)
```

```
## [1] 5000    4
```

```r
test4 = Default[indx,]
dim(test4)
```

```
## [1] 5000    4
```

```r
## model fitting
lr_model4 = glm(default ~ balance+income, data = train4, family = binomial)
print(contrasts(default))
```

```
##     Yes
## No    0
## Yes   1
```

```r
summary(lr_model4)
```

```
##
## Call:
## glm(formula = default ~ balance + income, family = binomial,
##     data = train4)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2421  -0.1352  -0.0532  -0.0199   3.7488
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.175e+01  6.183e-01 -18.996   <2e-16 ***
## balance      5.754e-03  3.247e-04  17.723   <2e-16 ***
## income       2.204e-05  7.167e-06   3.076   0.0021 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1450.21  on 4999  degrees of freedom
## Residual deviance:  755.93  on 4997  degrees of freedom
## AIC: 761.93
##
## Number of Fisher Scoring iterations: 8
```

```r
## error checking
pred_lr4 = predict(lr_model4,test4, type = 'response')
pred_class4 = ifelse(pred_lr4<0.5, 'No', 'Yes')
mean(test4$default != pred_class4)
```

```
## [1] 0.0276
```

The test errors for different splits are 0.023, 0.0316, 0.0276 repectively for 90-10, 75-25, 50-50 splitings. Therefore, it is clear that 90-10 spliting performs the best

### (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?

```r
# let us take 90-10 spliting, as it performed the best
set.seed(42)
n = dim(Default)[1]
indx = sample.int(n, floor(0.10*n))
train5 = Default[-indx,]
dim(train5)
```

```
## [1] 9000    4
```

```r
test5 = Default[indx,]
dim(test5)
```

```
## [1] 1000    4
```

```
## model fitting
lr_model5 = glm(default ~ balance+income, data = train5, family = binomial)
summary(lr_model5)
```

```
##
## Call:
## glm(formula = default ~ balance + income, family = binomial,
##     data = train5)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4738  -0.1426  -0.0567  -0.0208   3.7331
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.162e+01  4.602e-01 -25.254  < 2e-16 ***
## balance      5.652e-03  2.405e-04  23.495  < 2e-16 ***
## income       2.230e-05  5.274e-06   4.229 2.35e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2603.6  on 8999  degrees of freedom
## Residual deviance: 1403.7  on 8997  degrees of freedom
## AIC: 1409.7
##
## Number of Fisher Scoring iterations: 8
```

```
## error checking
pred_lr5 = predict(lr_model5,test5, type = 'response')
pred_class5 = ifelse(pred_lr5<0.5, 'No', 'Yes')
mean(test5$default != pred_class5)
```

```
## [1] 0.032
```

No, the test error has increased

# Problem 2

### 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)
```

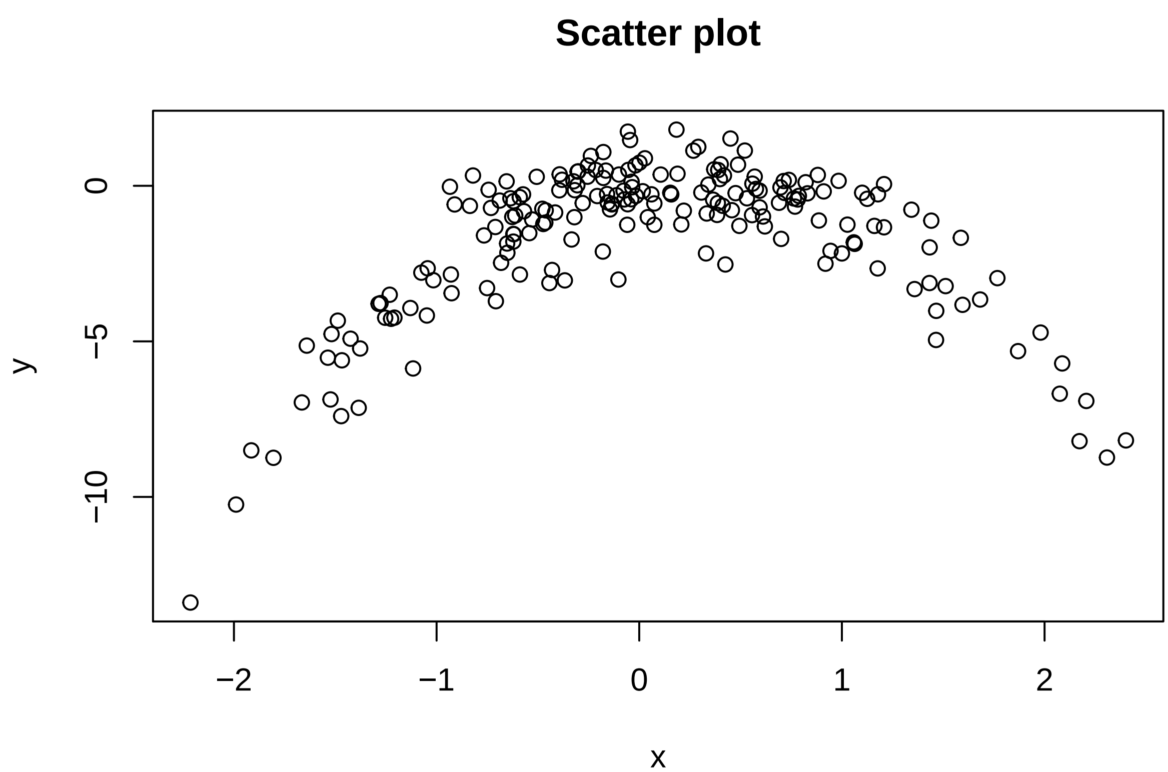
### In this data set, what is n???? and what pis ????? Write out the model used to generate the data in equation form (i.e., the true model of the data).

n = 200 p = 2 (without the intercept)

True model: Y = x - 2x^2 + e; e = error term

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

```r
plot(x,y, main = 'Scatter plot')
```




The relationship is clearly parabolic, which is obvious from the simulation. The values of Y increases with X, reaches maximum at about x = 0.5 and then reduces

### (c) Consider the following four models for the data set:

...

### Compute the LOOCV errors that result from fitting these models.

```r
#install.packages('RODBC')
suppressWarnings(suppressMessages(library(RODBC)))
options(warn = -1)
library(boot)
```

```r
data = data.frame(x,y)
lm_fit1 = glm(data$y~data$x, data=data)
loocv_glm1 = cv.glm(data, lm_fit1)
loocv_glm1$delta
```

```
## [1] 6.164095 6.164095
```

```r
# note: 2 deltas - 1st is raw cross-validation, 2nd is bias corrected (when LOOCV not used, a bias is i

data = data.frame(x,y)
lm_fit2 = glm(data$y~data$x+I(data$x^2), data=data)
loocv_glm2 = cv.glm(data, lm_fit2)
loocv_glm2$delta
```

```
## [1] 10.93846 10.93846
```

```r
lm_fit3 = glm(data$y~data$x+I(data$x^2)+I(data$x^3), data=data)
loocv_glm3 = cv.glm(data, lm_fit3)
loocv_glm3$delta
```

```
## [1] 10.95005 10.95005
```

```r
lm_fit4 = glm(data$y~data$x+I(data$x^2)+I(data$x^3)+I(data$x^4), data=data)
loocv_glm4 = cv.glm(data, lm_fit4)
loocv_glm4$delta
```

```
## [1] 10.96536 10.96536
```

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

```r
set.seed(101)
x=rnorm(200)
y=x-2*x^2+rnorm(200)
data = data.frame(x,y)
lm_fit1 = glm(data$y~data$x, data=data)
loocv_glm1 = cv.glm(data, lm_fit1)
loocv_glm1$delta
```

```
## [1] 11.36976 11.36976
```

```r
# note: 2 deltas - 1st is raw cross-validation, 2nd is bias corrected (when LOOCV not used, a bias is i

data = data.frame(x,y)
lm_fit2 = glm(data$y~data$x+I(data$x^2), data=data)
loocv_glm2 = cv.glm(data, lm_fit2)
loocv_glm2$delta
```

```
## [1] 18.03395 18.03395
```

```r
lm_fit3 = glm(data$y~data$x+I(data$x^2)+I(data$x^3), data=data)
loocv_glm3 = cv.glm(data, lm_fit3)
loocv_glm3$delta
```

```
## [1] 18.04254 18.04254
```

```
lm_fit4 = glm(data$y~data$x+I(data$x^2)+I(data$x^3)+I(data$x^4), data=data)
loocv_glm4 = cv.glm(data, lm_fit4)
loocv_glm4$delta
```

```
## [1] 18.04502 18.04502
```

The results are different.

This is because, by changing the seed, we change the values of randomly generated x and corresponding y values. Therefore, the fit changes and hence different results are obtained

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

The first model has the smallest LOOCV

No, I did not expect it to be smallest. But, it explains the non-linear models overfit the data and hence the validation error increases

### (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?

```
lm_fit1 = glm(data$y~data$x, data=data)
cv_glm1 = cv.glm(data, lm_fit1, K=5)
cv_glm1$delta
```

```
## [1] 11.57795 11.57795
```

```
lm_fit2 = glm(data$y~data$x+I(data$x^2), data=data)
cv_glm2 = cv.glm(data, lm_fit2, K=5)
cv_glm2$delta
```

```
## [1] 18.83344 18.83344
```

```
lm_fit3 = glm(data$y~data$x+I(data$x^2)+I(data$x^3), data=data)
cv_glm3 = cv.glm(data, lm_fit3, K=5)
cv_glm3$delta
```

```
## [1] 18.47237 18.47237
```

```
lm_fit4 = glm(data$y~data$x+I(data$x^2)+I(data$x^3)+I(data$x^4), data=data)
cv_glm4 = cv.glm(data, lm_fit4, K=5)
cv_glm4$delta
```

```
## [1] 18.2444 18.2444
```

Again, first model has lowest error.

5 fold cross validation shows that higher degree polynomials overfit the data and the error increases.

The result is inline with the LOOCV errors, but corresponding errors are higher than LOOCV

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

```r
lm_fit1 = glm(data$y~data$x, data=data)
loocv_glm1 = cv.glm(data, lm_fit1, K=10)
loocv_glm1$delta
```

```
## [1] 11.07196 11.07196
```

```r
lm_fit2 = glm(data$y~data$x+I(data$x^2), data=data)
loocv_glm2 = cv.glm(data, lm_fit2, K=10)
loocv_glm2$delta
```

```
## [1] 18.36473 18.36473
```

```r
lm_fit3 = glm(data$y~data$x+I(data$x^2)+I(data$x^3), data=data)
loocv_glm3 = cv.glm(data, lm_fit3, K=10)
loocv_glm3$delta
```

```
## [1] 18.02551 18.02551
```

```r
lm_fit4 = glm(data$y~data$x+I(data$x^2)+I(data$x^3)+I(data$x^4), data=data)
loocv_glm4 = cv.glm(data, lm_fit4, K=10)
loocv_glm4$delta
```

```
## [1] 19.05525 19.05525
```

Again, first model has lowest error.

10 fold cross validation shows that higher degree polynomials overfit the data and the error increases.

The result is inline with the LOOCV & 5-fold errors, but corresponding errors are higher