Li\_Tianyu\_Sun\_Jingyan\_HW2

Tianyu Li and Jingyan Sun

February 19, 2019

# R Markdown

# Question 1(Cha4-Q1)

Proof:

From (4.2)

We can get

So, we have proofed that equation 4.3 (odds) can be deduced from equation 4.2 (logistic function).

# Question 2(Cha4-Q5)

(a) If the Bayes decision boundary is linear, we expect QDA to perform better on training set because of its higher flexibility; while LDA perform better on test set than QLA does, because QDA could overfit the linearity on the Bayes decision boundary.

(b) If the Bayes decision boundary is non-linear, we expect QDA better on both.

(c) General speaking, LDA tends to be a better bet than QDA if there are relatively few training observations. In contrast, when sample size n (training set) increases, QDA is recommended, so that the variance of the classifier is not a major concern, or if the assumption of a common covariance matrix for the K classes is clearly.

(d) False. It depends on different situation, sometimes it is an advantage to apply a method with more flexibility such as QDA which may lead a superior test error rate, but it can turn to a disadvantage when too much flexible may lead to overfit, turning to an inferior test error rate.

# Question 3(Cha4-Q8)

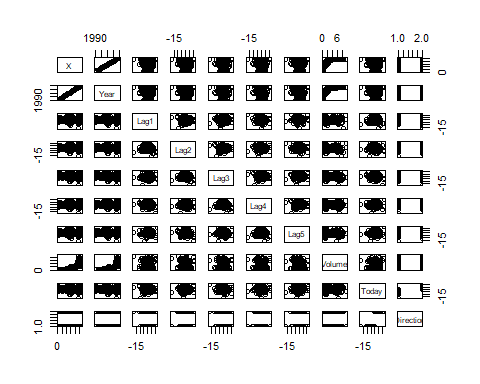
In this case, we should choose logistic regression because of its lower test error rate. With KNN with K=1, the training error rate is always 0%. Given the average error rate (averaged over both test and training data sets) of it is 18%, we can see that the tests error rate is 36%. On the opposite, by using logistic regression, the test error rate is 30%, which is smaller than the test error rate of KNN when K=1. So, it is more appropriate to choose logistic regression based on the test error rate.

# Question 4(Cha4-Q10)

# Author: Tianyu Li  
# Created on Feb 18th, 2019  
#  
# R script for Homework 2 Question 4(Section 4.7, page 171, question 10)  
# The College.csv file should be in working direction   
rm(list = ls())  
setwd('Z:/R\_working\_directory/DS502HW2');  
library(MASS)  
library(class)  
  
# Read the file and set the random seed  
ds = read.csv(file = 'weekly.csv', header = TRUE);  
set.seed(1)  
  
# (a) Produce some numerical and graphical summaries of the Weekly data  
summary(ds)

## X Year Lag1 Lag2   
## Min. : 1 Min. :1990 Min. :-18.1950 Min. :-18.1950   
## 1st Qu.: 273 1st Qu.:1995 1st Qu.: -1.1540 1st Qu.: -1.1540   
## Median : 545 Median :2000 Median : 0.2410 Median : 0.2410   
## Mean : 545 Mean :2000 Mean : 0.1506 Mean : 0.1511   
## 3rd Qu.: 817 3rd Qu.:2005 3rd Qu.: 1.4050 3rd Qu.: 1.4090   
## Max. :1089 Max. :2010 Max. : 12.0260 Max. : 12.0260   
## Lag3 Lag4 Lag5   
## Min. :-18.1950 Min. :-18.1950 Min. :-18.1950   
## 1st Qu.: -1.1580 1st Qu.: -1.1580 1st Qu.: -1.1660   
## Median : 0.2410 Median : 0.2380 Median : 0.2340   
## Mean : 0.1472 Mean : 0.1458 Mean : 0.1399   
## 3rd Qu.: 1.4090 3rd Qu.: 1.4090 3rd Qu.: 1.4050   
## Max. : 12.0260 Max. : 12.0260 Max. : 12.0260   
## Volume Today Direction   
## Min. :0.08747 Min. :-18.1950 Down:484   
## 1st Qu.:0.33202 1st Qu.: -1.1540 Up :605   
## Median :1.00268 Median : 0.2410   
## Mean :1.57462 Mean : 0.1499   
## 3rd Qu.:2.05373 3rd Qu.: 1.4050   
## Max. :9.32821 Max. : 12.0260

pairs(ds)



# From the figures we can tell that there appears to be   
# relationship between Year and Volume: Volume is increasing  
# as year increasing  
  
# (b) Perform a logistic regression with Direction over Lags and Volume  
fit = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,  
 family = "binomial", data = ds)  
summary(fit)

##   
## Call:  
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +   
## Volume, family = "binomial", data = ds)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.6949 -1.2565 0.9913 1.0849 1.4579   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.26686 0.08593 3.106 0.0019 \*\*  
## Lag1 -0.04127 0.02641 -1.563 0.1181   
## Lag2 0.05844 0.02686 2.175 0.0296 \*   
## Lag3 -0.01606 0.02666 -0.602 0.5469   
## Lag4 -0.02779 0.02646 -1.050 0.2937   
## Lag5 -0.01447 0.02638 -0.549 0.5833   
## Volume -0.02274 0.03690 -0.616 0.5377   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1496.2 on 1088 degrees of freedom  
## Residual deviance: 1486.4 on 1082 degrees of freedom  
## AIC: 1500.4  
##   
## Number of Fisher Scoring iterations: 4

# It looks like only Lag2 appear to be statistically significant as  
# its p-value smaller than 0.05  
  
# (c) Compute the confusion matrix and fraction of correct predictions  
pred = predict(fit, type = "response")  
result = rep("Down", length(pred))  
result[pred > 0.5] = "Up"  
  
# Confusion matrix  
table(result, ds$Direction)

##   
## result Down Up  
## Down 54 48  
## Up 430 557

# Fraction of correct predictions  
mean(result == ds$Direction)

## [1] 0.5610652

# The confusion matrix shows that the model tends to predict most directions  
# as "UP", which leads to a only 56.1% accuracy on prediction.  
  
# (d) Perform a logistic regression with Lag2 as the predictor  
  
# Helper function to calculate the fraction of correct predictions from   
# the confusion matrix  
accuracy = function(table) {  
 result = (table[1, 1] + table[2, 2]) / sum(table)  
 return (result)  
}  
  
fit = glm(Direction ~ Lag2, family = "binomial",  
 data = ds, subset = Year < 2009)  
fit

##   
## Call: glm(formula = Direction ~ Lag2, family = "binomial", data = ds,   
## subset = Year < 2009)  
##   
## Coefficients:  
## (Intercept) Lag2   
## 0.2033 0.0581   
##   
## Degrees of Freedom: 984 Total (i.e. Null); 983 Residual  
## Null Deviance: 1355   
## Residual Deviance: 1351 AIC: 1355

# Confusion matrix and fraction of correct predictions  
pred = predict.glm(fit, subset(ds, Year >= 2009), type = "response")  
result = rep("Down", length(pred))  
result[pred > 0.5] = "Up"  
d\_table = table(result, subset(ds, Year >= 2009)$Direction)  
d\_table

##   
## result Down Up  
## Down 9 5  
## Up 34 56

accuracy(d\_table)

## [1] 0.625

# (e) Perform a LDA with Lag2 as the predictor  
fit = lda(Direction ~ Lag2, data = ds, subset = Year < 2009)  
fit

## Call:  
## lda(Direction ~ Lag2, data = ds, subset = Year < 2009)  
##   
## Prior probabilities of groups:  
## Down Up   
## 0.4477157 0.5522843   
##   
## Group means:  
## Lag2  
## Down -0.03568254  
## Up 0.26036581  
##   
## Coefficients of linear discriminants:  
## LD1  
## Lag2 0.4414162

# Confusion matrix and fraction of correct predictions  
pred = predict(fit, subset(ds, Year >= 2009), type = "response")  
e\_table = table(pred$class, subset(ds, Year >= 2009)$Direction)  
e\_table

##   
## Down Up  
## Down 9 5  
## Up 34 56

accuracy(e\_table)

## [1] 0.625

# (f) Perform a QDA Lag2 as the predictor  
fit = qda(Direction ~ Lag2, data = ds, subset = Year < 2009)  
fit

## Call:  
## qda(Direction ~ Lag2, data = ds, subset = Year < 2009)  
##   
## Prior probabilities of groups:  
## Down Up   
## 0.4477157 0.5522843   
##   
## Group means:  
## Lag2  
## Down -0.03568254  
## Up 0.26036581

# Confusion matrix and fraction of correct predictions  
pred = predict(fit, subset(ds, Year >= 2009), type = "response")  
f\_table = table(pred$class, subset(ds, Year >= 2009)$Direction)  
f\_table

##   
## Down Up  
## Down 0 0  
## Up 43 61

accuracy(f\_table)

## [1] 0.5865385

# (g) Perform a KNN with Lag2 as the predictor  
pred = knn(data.frame(subset(ds, Year < 2009)$Lag2),  
 data.frame(subset(ds, Year >= 2009)$Lag2),  
 subset(ds, Year < 2009)$Direction, k = 1)  
  
# Confusion matrix and fraction of correct predictions  
g\_table = table(pred, subset(ds, Year >= 2009)$Direction)  
g\_table

##   
## pred Down Up  
## Down 21 30  
## Up 22 31

accuracy(g\_table)

## [1] 0.5

# (h) Compare results  
accuracy(d\_table)

## [1] 0.625

accuracy(e\_table)

## [1] 0.625

accuracy(f\_table)

## [1] 0.5865385

accuracy(g\_table)

## [1] 0.5

# In our test, Logistic regression and LDA give the highest   
# fraction of correct predictions, then QDA,   
# and KNN with k=1 has the lowest accuracy rate  
  
# (i) Experiments  
# logistic Regression  
fit = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,  
 family = "binomial", data = ds, subset = Year < 2009)  
pred = predict.glm(fit, subset(ds, Year >= 2009), type = "response")  
result = rep("Down", length(pred))  
result[pred > 0.5] = "Up"  
glm1 = table(result, subset(ds, Year >= 2009)$Direction)  
  
fit = glm(Direction ~ Lag1 \* Lag2 \* Lag3 \* Lag4 \* Lag5 + Volume,  
 family = "binomial", data = ds, subset = Year < 2009)

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

pred = predict.glm(fit, subset(ds, Year >= 2009), type = "response")  
result = rep("Down", length(pred))  
result[pred > 0.5] = "Up"  
glm2 = table(result, subset(ds, Year >= 2009)$Direction)  
  
# LDA  
fit = lda(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,  
 data = ds, subset = Year < 2009)  
pred = predict(fit, subset(ds, Year >= 2009), type = "response")  
lda1 = table(pred$class, subset(ds, Year >= 2009)$Direction)  
  
fit = lda(Direction ~ Lag1 \* Lag2 \* Lag3 \* Lag4 \* Lag5 + Volume,  
 data = ds, subset = Year < 2009)  
pred = predict(fit, subset(ds, Year >= 2009), type = "response")  
lda2 = table(pred$class, subset(ds, Year >= 2009)$Direction)  
  
# QDA  
fit = qda(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,  
 data = ds, subset = Year < 2009)  
pred = predict(fit, subset(ds, Year >= 2009), type = "response")  
qda1 = table(pred$class, subset(ds, Year >= 2009)$Direction)  
  
fit = qda(Direction ~ Lag1 \* Lag2 \* Lag3 \* Lag4 \* Lag5 + Volume,  
 data = ds, subset = Year < 2009)  
pred = predict(fit, subset(ds, Year >= 2009), type = "response")  
qda2 = table(pred$class, subset(ds, Year >= 2009)$Direction)  
  
#KNN  
pred = knn(data.frame(subset(ds, Year < 2009)$Lag2),  
 data.frame(subset(ds, Year >= 2009)$Lag2),  
 subset(ds, Year < 2009)$Direction, k = 10)  
knn1 = table(pred, subset(ds, Year >= 2009)$Direction)  
  
pred = knn(data.frame(subset(ds, Year < 2009)$Lag2),  
 data.frame(subset(ds, Year >= 2009)$Lag2),  
 subset(ds, Year < 2009)$Direction, k = 50)  
knn2 = table(pred, subset(ds, Year >= 2009)$Direction)  
  
accuracy(glm1)

## [1] 0.4615385

accuracy(glm2)

## [1] 0.4519231

accuracy(lda1)

## [1] 0.4615385

accuracy(lda2)

## [1] 0.4326923

accuracy(qda1)

## [1] 0.4326923

accuracy(qda2)

## [1] 0.4134615

accuracy(knn1)

## [1] 0.5576923

accuracy(knn2)

## [1] 0.5865385

accuracy(d\_table)

## [1] 0.625

accuracy(e\_table)

## [1] 0.625

accuracy(f\_table)

## [1] 0.5865385

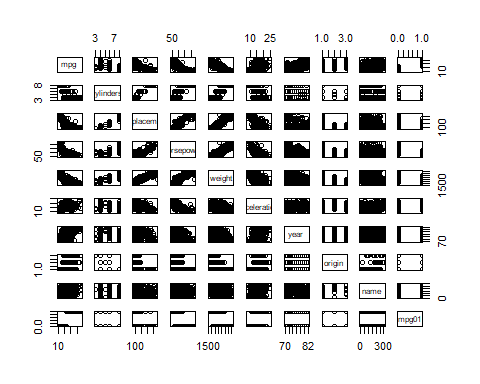
accuracy(g\_table)

## [1] 0.5

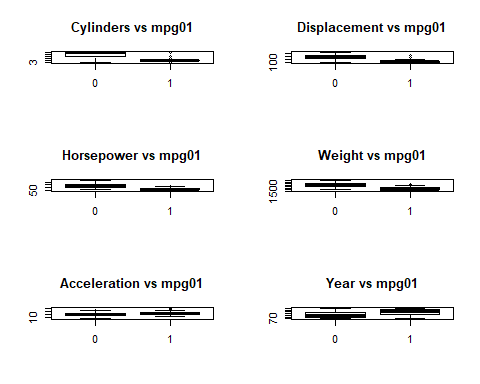
# THe K value in KNN does not seem to influcence the test error rate significantly  
# And over all experiments, the original logsitic regression with Volume over lag2,  
# and LDA with Volume over lag2 still have the lowest error rate.

# Question 5(Cha4-Q11)

# Author: Tianyu Li  
# Created on Feb 18th, 2019  
#  
# R script for Homework 2 Question 5(Section 4.7, page 171, question 11)  
# The College.csv file should be in working direction   
rm(list = ls())  
setwd('Z:/R\_working\_directory/DS502HW2');  
library(MASS)  
library(class)  
  
# Read the file and set the random seed  
ds = read.csv(file = 'Auto.csv', header = TRUE);  
set.seed(2)  
  
# Remove missing values  
ds[ds == '?'] <- NA;  
ds = na.omit(ds);  
ds$horsepower = as.numeric(as.character(ds$horsepower));  
  
# (a) Create variable mpg01  
mpg01 = rep(0, nrow(ds))  
mpg01[ds$mpg > median(ds$mpg)] = 1  
ds = data.frame(ds, mpg01)  
  
# (b) Explore data and investigate the association   
# between "mpg01" and the other features  
par(mfrow=c(1,1))  
pairs(ds)



par(mfrow=c(3,2))  
boxplot(cylinders ~ mpg01, data = ds, main = "Cylinders vs mpg01")  
boxplot(displacement ~ mpg01, data = ds, main = "Displacement vs mpg01")  
boxplot(horsepower ~ mpg01, data = ds, main = "Horsepower vs mpg01")  
boxplot(weight ~ mpg01, data = ds, main = "Weight vs mpg01")  
boxplot(acceleration ~ mpg01, data = ds, main = "Acceleration vs mpg01")  
boxplot(year ~ mpg01, data = ds, main = "Year vs mpg01")



# It looks like that cylinders, displacement, horse power and weight have  
# Strong association with mpg01, and acceleration has weak association.  
  
# (c) Split the data  
index = sample(nrow(ds), nrow(ds)/2)  
train = ds[index, ]  
test = ds[-index, ]  
  
# (d) Pefrome LDA  
fit = lda(mpg01 ~ cylinders + displacement + horsepower + weight + acceleration,  
 data = train)  
fit

## Call:  
## lda(mpg01 ~ cylinders + displacement + horsepower + weight +   
## acceleration, data = train)  
##   
## Prior probabilities of groups:  
## 0 1   
## 0.4693878 0.5306122   
##   
## Group means:  
## cylinders displacement horsepower weight acceleration  
## 0 6.891304 280.1739 135.69565 3609.000 13.88478  
## 1 4.096154 112.4519 77.10577 2294.635 16.72596  
##   
## Coefficients of linear discriminants:  
## LD1  
## cylinders -8.013196e-01  
## displacement 4.548390e-03  
## horsepower -6.255509e-05  
## weight -8.802328e-04  
## acceleration 4.484453e-02

# Test error  
pred = predict(fit, test, type = "response")  
table(pred$class, test$mpg01)

##   
## 0 1  
## 0 90 10  
## 1 14 82

mean(pred$class != test$mpg01)

## [1] 0.122449

# (e) Pefrome QDA  
fit = qda(mpg01 ~ cylinders + displacement + horsepower + weight + acceleration,  
 data = train)  
fit

## Call:  
## qda(mpg01 ~ cylinders + displacement + horsepower + weight +   
## acceleration, data = train)  
##   
## Prior probabilities of groups:  
## 0 1   
## 0.4693878 0.5306122   
##   
## Group means:  
## cylinders displacement horsepower weight acceleration  
## 0 6.891304 280.1739 135.69565 3609.000 13.88478  
## 1 4.096154 112.4519 77.10577 2294.635 16.72596

# Test error  
pred = predict(fit, test, type = "response")  
table(pred$class, test$mpg01)

##   
## 0 1  
## 0 91 11  
## 1 13 81

mean(pred$class != test$mpg01)

## [1] 0.122449

# (f) Perform logistic regression  
fit = glm(mpg01 ~ cylinders + displacement + horsepower + weight + acceleration,  
 family = "binomial", data = train)  
fit

##   
## Call: glm(formula = mpg01 ~ cylinders + displacement + horsepower +   
## weight + acceleration, family = "binomial", data = train)  
##   
## Coefficients:  
## (Intercept) cylinders displacement horsepower weight   
## 14.522649 -1.057951 0.005736 -0.058218 -0.002348   
## acceleration   
## 0.113055   
##   
## Degrees of Freedom: 195 Total (i.e. Null); 190 Residual  
## Null Deviance: 271   
## Residual Deviance: 85.16 AIC: 97.16

# Test error  
pred = predict.glm(fit, test, type = "response")  
result = rep(0, length(pred))  
result[pred > 0.5] = 1  
table(result, test$mpg01)

##   
## result 0 1  
## 0 91 13  
## 1 13 79

mean(result != test$mpg01)

## [1] 0.1326531

# (g) Perform KNN  
# K = 1  
pred = knn(data.frame(train$cylinders, train$displacement, train$horsepower, train$weight, train$acceleration),  
 data.frame(test$cylinders, test$displacement, test$horsepower, test$weight, test$acceleration),  
 train$mpg01, k = 1)  
table(pred, test$mpg01)

##   
## pred 0 1  
## 0 91 17  
## 1 13 75

mean(pred != test$mpg01)

## [1] 0.1530612

# K = 5  
pred = knn(data.frame(train$cylinders, train$displacement, train$horsepower, train$weight, train$acceleration),  
 data.frame(test$cylinders, test$displacement, test$horsepower, test$weight, test$acceleration),  
 train$mpg01, k = 5)  
table(pred, test$mpg01)

##   
## pred 0 1  
## 0 91 12  
## 1 13 80

mean(pred != test$mpg01)

## [1] 0.127551

# K = 20  
pred = knn(data.frame(train$cylinders, train$displacement, train$horsepower, train$weight, train$acceleration),  
 data.frame(test$cylinders, test$displacement, test$horsepower, test$weight, test$acceleration),  
 train$mpg01, k = 20)  
table(pred, test$mpg01)

##   
## pred 0 1  
## 0 92 14  
## 1 12 78

mean(pred != test$mpg01)

## [1] 0.1326531

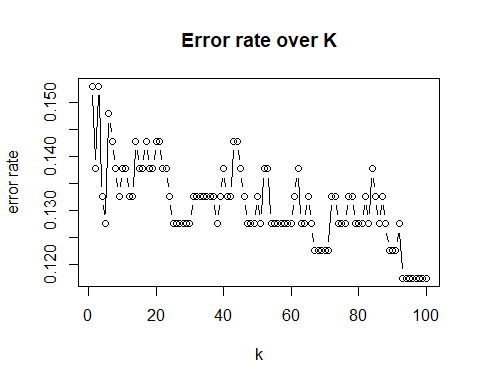
# K = 100  
pred = knn(data.frame(train$cylinders, train$displacement, train$horsepower, train$weight, train$acceleration),  
 data.frame(test$cylinders, test$displacement, test$horsepower, test$weight, test$acceleration),  
 train$mpg01, k = 100)  
table(pred, test$mpg01)

##   
## pred 0 1  
## 0 86 6  
## 1 18 86

mean(pred != test$mpg01)

## [1] 0.122449

# Error rate over K  
acc\_knn = function(k) {  
 pred = knn(data.frame(train$cylinders, train$displacement, train$horsepower, train$weight, train$acceleration),  
 data.frame(test$cylinders, test$displacement, test$horsepower, test$weight, test$acceleration),  
 train$mpg01, k = k)  
 return (mean(pred != test$mpg01))  
}  
  
x = vector()  
y = vector()  
for(i in 1:100) {  
 x[i] = i  
 y[i] = acc\_knn(i)  
}  
  
par(mfrow=c(1,1))  
plot(x, y, type = "b", xlab = "k", ylab = "error rate", main = "Error rate over K")



# It looks like K does not influence the performance on this data set.

# Question 6(Cha5-Q1)

Proof:

Take the first derivative of Var related to , we can get:

To check whether the result is minimum, we can take the second derivative of , checking whether it greater than or equal to zero:

# Question 7(Cha5-Q2)

(a) 1-1/n.

Because bootstrap sampling draws items with replacement, we are sampling from the same pool with the same probability for each time, so there are (n-1) items in the n that are not j and with a probability of 1-1/n that the first item is not j.

(b) 1-1/n

Still because the bootstrap sampling pattern is with replacement, so the probability would not change in this case.

(c)

In previous situation, we can get the probability that the jth observation is not in the bootstrap sample is (1-1/n) for each time sampling. With the bootstrap where n!=j, the same situation applies and probability for j have to repeated for n times, then the result would be .

(d)

When N=5,

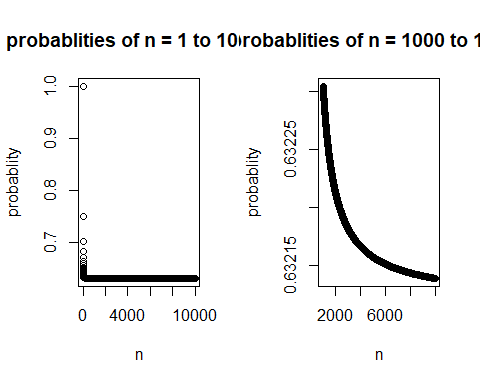
(e)

When N=100,

(f)

When N=10000,

# Author: Tianyu Li  
# Created on Feb 18th, 2019  
#  
# R script for Homework 2 Question 7(Section 5.4, page 198, question 8)  
rm(list = ls())  
  
#(g) Create a plot on probablities of n = 1 to 10000  
x = vector()  
y = vector()  
  
for(n in 1:10000) {  
 x[n] = n  
 y[n] = 1 - ((1 - 1/n) ^ n)  
}  
  
par(mfrow=c(1,2))  
plot(x, y, xlab = "n", ylab = "probablity",  
 main = "probablities of n = 1 to 10000")  
plot(x[1000:10000], y[1000:10000], xlab = "n", ylab = "probablity",  
 main = "probablities of n = 1000 to 10000")



# The probablity that that the jth observation is in the bootstrap sample  
# is decreasing as the size of observation n increasing. And it looks like  
# The probablity converges to around 0.63  
  
# (h) Probablity on j = 4 by bootstrap samples  
results = vector()  
for(i in 1:10) {  
store = rep(NA, 10000)  
 for(n in 1:10000) {  
 store[n] = sum(sample(1:100, rep = TRUE) == 4) > 0  
 }  
 results[i] = mean(store)  
}  
results

## [1] 0.6313 0.6271 0.6300 0.6365 0.6398 0.6388 0.6383 0.6403 0.6342 0.6348

# The probablity is about 0.63, which shows that the result  
# we got on theory is correct

# Question 8(Cha5-Q5)

# Author: Tianyu Li  
# Created on Feb 18th, 2019  
#  
# R script for Homework 2 Question 8(Section 5.4, page 198, question 5)  
# The College.csv file should be in working direction   
rm(list = ls())  
setwd('Z:/R\_working\_directory/DS502HW2');  
  
# Read the file and set the random seed  
ds = read.csv(file = 'default.csv', header = TRUE);  
set.seed(3)  
  
# (a) Fit a logistic regression model that uses income and balance  
# to predict default  
fit = glm(default ~ income + balance, family = binomial, data = ds)  
summary(fit)

##   
## Call:  
## glm(formula = default ~ income + balance, family = binomial,   
## data = ds)  
##   
## 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  
# i. Split the sample set  
train = sample(nrow(ds), nrow(ds)/2)  
  
# ii. Fit with training set  
train\_fit = glm(default ~ income + balance, family = binomial,  
 data = ds, subset = train)  
summary(train\_fit)

##   
## Call:  
## glm(formula = default ~ income + balance, family = binomial,   
## data = ds, subset = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.1014 -0.1433 -0.0569 -0.0206 3.7241   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.160e+01 6.055e-01 -19.162 < 2e-16 \*\*\*  
## income 2.254e-05 6.972e-06 3.233 0.00123 \*\*   
## balance 5.660e-03 3.131e-04 18.079 < 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: 1530.39 on 4999 degrees of freedom  
## Residual deviance: 812.77 on 4997 degrees of freedom  
## AIC: 818.77  
##   
## Number of Fisher Scoring iterations: 8

# iii. Validate the fit with testing set and classify with   
# if the posterior probability is greater than 0.5.  
pred = predict.glm(train\_fit, newdata = ds[-train, ], type = 'response')  
result = rep("No", length(pred))  
result[pred > 0.5] = "Yes"  
  
# iv Compte the validation set error  
mean(result != ds[-train, ]$default)

## [1] 0.0248

# (c) Repeat the process in (b) three times  
# Define the process as a function  
error = function(){  
 train = sample(nrow(ds), nrow(ds)/2)  
 train\_fit = glm(default ~ income + balance, family = binomial,  
 data = ds, subset = train)  
 summary(train\_fit)  
 pred = predict.glm(train\_fit, newdata = ds[-train, ], type = 'response')  
 result = rep("No", length(pred))  
 result[pred > 0.5] = "Yes"  
   
 error = mean(result != ds[-train, ]$default)  
 return(error)  
}  
  
# Repeat the function three times  
error()

## [1] 0.0258

error()

## [1] 0.0242

error()

## [1] 0.0264

# The test error rate could be floating as the sample process is random  
  
# (d) Add a dummy variable for student and test  
error2 = function(){  
 train = sample(nrow(ds), nrow(ds)/2)  
 train\_fit = glm(default ~ income + balance + student, family = binomial,  
 data = ds, subset = train)  
 summary(train\_fit)  
 pred = predict.glm(train\_fit, newdata = ds[-train, ], type = 'response')  
 result = rep("No", length(pred))  
 result[pred > 0.5] = "Yes"  
   
 error = mean(result != ds[-train, ]$default)  
 return(error)  
}  
  
error2()

## [1] 0.0258

error2()

## [1] 0.0262

error2()

## [1] 0.0236

# It does not seems like adding a dummy variable for "student" would  
# reduce the test error rate.

# Question 9(Cha5-Q6)

# Author: Tianyu Li  
# Created on Feb 18th, 2019  
#  
# R script for Homework 2 Question 9(Section 5.4, page 199, question 6)  
# The College.csv file should be in working direction   
rm(list = ls())  
setwd('Z:/R\_working\_directory/DS502HW2');  
library(boot)  
  
# Read the file and set the random seed  
ds = read.csv(file = 'default.csv', header = TRUE);  
set.seed(4)  
  
# (a) Determine the estimated standard errors  
fit = glm(default ~ income + balance, family = "binomial",  
 data = ds)  
summary(fit)

##   
## Call:  
## glm(formula = default ~ income + balance, family = "binomial",   
## data = ds)  
##   
## 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) Write boot function  
boot\_fn = function(data, index) {  
 fit = glm(default ~ income + balance, family = "binomial",  
 data = data, subset = index)  
 return (coef(fit))  
}  
  
# (c) test with bot function  
boot(ds, boot\_fn, 1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = ds, statistic = boot\_fn, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* -1.154047e+01 -2.107293e-02 4.335119e-01  
## t2\* 2.080898e-05 -2.398464e-07 4.671074e-06  
## t3\* 5.647103e-03 1.499666e-05 2.262465e-04

# (d) Comments  
# It looks like the estimated standard errors obtained by  
# these two methods are pretty close

Note that the echo = FALSE parameter was added to the code chunk to prevent printing of the R code that generated the plot.