Homework_3

Akeem Ajede

10/21/2019

Question 5 - Logistic Regression

Q5(a)

Fitting a logistic regression model.

```
library(ISLR)
library(MASS)
RNGkind(sample.kind = "Rounding") #To correct the RNG of different R-studio versions
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
set.seed(1)
attach(Default)
fit.glm1.0 <- glm(default~income+balance,
               data = Default, family = binomial)
summary(fit.glm1.0)$coef #both predictors are statistically significant
```

```
             Estimate   Std. Error    z value      Pr(>|z|)
```

(Intercept) -1.154047e+01 4.347564e-01 -26.544680 2.958355e-155 income 2.080898e-05 4.985167e-06 4.174178 2.990638e-05 balance 5.647103e-03 2.273731e-04 24.836280 3.638120e-136

Q5(b)

Validation exercise.

```
## Sample splitting
set.seed(1)
smpl.size <- floor(0.5*nrow(Default))
train <- sample(seq_len(nrow(Default)), size = smpl.size)
def.train <- Default[train, ]
def.test <- Default[-train, ]
def.c<-default[-train]

## Fitting Multiple Logistic Regression
fit.glm1.1<-glm(default~income+balance,
                data = Default, family = binomial, subset = train)
prob.glm<-predict(fit.glm1.1, def.test, type = "response")
pred.glm<-rep("No", 5000)
pred.glm[prob.glm>.5]<-"Yes"
table(pred.glm,def.c)
```

```
    def.c
```

pred.glm No Yes No 4805 115 Yes 28 52

```
mean(pred.glm==def.c) #model accuracy is 97.1%
```

[1] 0.9714

```
mean(pred.glm!= def.c) #The test (validation) error is 0.0286
```

[1] 0.0286

At the granular level, the confusion matrix suggests that the actual "default" miscalculation is approx. 68.9%! such poor precision level may be unacceptable to a credit card company.

Q5(c)

validation exercise w/ different seeds.

```
set.seed(2)
smpl.size <- floor(0.5*nrow(Default))
train <- sample(seq_len(nrow(Default)), size = smpl.size)
def.train <- Default[train, ]
def.test <- Default[-train, ]
def.c<-default[-train]
fit.glm1.1<-glm(default~income+balance,
                data = Default, family = binomial, subset = train)
prob.glm<-predict(fit.glm1.1, def.test, type = "response")
pred.glm<-rep("No", 5000)
pred.glm[prob.glm>.5]<-"Yes"
table(pred.glm,def.c)
```

```
    def.c
```

pred.glm No Yes No 4811 118 Yes 20 51

```
mean(pred.glm!= def.c) #The test (validation) error is 0.0276
```

[1] 0.0276

```
set.seed(3)
smpl.size <- floor(0.5*nrow(Default))
train <- sample(seq_len(nrow(Default)), size = smpl.size)
def.train <- Default[train, ]
def.test <- Default[-train, ]
def.c<-default[-train]
fit.glm1.1<-glm(default~income+balance,
                data = Default, family = binomial, subset = train)
prob.glm<-predict(fit.glm1.1, def.test, type = "response")
pred.glm<-rep("No", 5000)
pred.glm[prob.glm>.5]<-"Yes"
table(pred.glm,def.c)
```

```
    def.c
```

pred.glm No Yes No 4828 108 Yes 16 48

```
mean(pred.glm!= def.c) #The test (validation) error is 0.0248
```

[1] 0.0248

```
set.seed(4)
smpl.size <- floor(0.5*nrow(Default))
train <- sample(seq_len(nrow(Default)), size = smpl.size)
def.train <- Default[train, ]
def.test <- Default[-train, ]
def.c<-default[-train]
fit.glm1.1<-glm(default~income+balance,
                data = Default, family = binomial, subset = train)
prob.glm<-predict(fit.glm1.1, def.test, type = "response")
pred.glm<-rep("No", 5000)
pred.glm[prob.glm>.5]<-"Yes"
table(pred.glm,def.c)
```

```
    def.c
```

pred.glm No Yes No 4813 112 Yes 19 56

```
mean(pred.glm!= def.c) #The test (validation) error is 0.0262
```

[1] 0.0262

The test error was different for the three different sample splits.

Q5(d)

Addition of the dummy variable "student."

```
set.seed(1)
smpl.size <- floor(0.5*nrow(Default))
train <- sample(seq_len(nrow(Default)), size = smpl.size)
def.train <- Default[train, ]
def.test <- Default[-train, ]
def.c<-default[-train]
fit.glm1.2<-glm(default~income+balance+student,
                data = Default, family = binomial, subset = train)
prob.glm<-predict(fit.glm1.2, def.test, type = "response")
pred.glm <- ifelse(prob.glm > 0.5, "Yes", "No")
table(pred.glm,def.c)
```

```
    def.c
```

pred.glm No Yes No 4803 114 Yes 30 53

```r
mean(pred.glm!= def.c)#The test (validation) error is 0.0288
```

[1] 0.0288

Question 6 - Logistic Regression; Coefficients estimate computation

Q6(a)

standard errors of coefficients.

```
set.seed(1)
smpl.size <- floor(0.5*nrow(Default))
train <- sample(seq_len(nrow(Default)), size = smpl.size)
def.train <- Default[train, ]
def.test <- Default[-train, ]
def.c<-default[-train]
fit.glm1.1<-glm(default~income+balance,
                data = Default, family = binomial, subset = train)
summary(fit.glm1.1)
```

Call: glm(formula = default ~ income + balance, family = binomial, data = Default, subset = train)

Deviance Residuals: Min 1Q Median 3Q Max
-2.3583 -0.1268 -0.0475 -0.0165 3.8116

Coefficients: Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.208e+01 6.658e-01 -18.148 <2e-16 ***income 1.858e-05 7.573e-06 2.454 0.0141***
balance 6.053e-03 3.467e-04 17.457 <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: 1457.0  on 4999  degrees of freedom
```

Residual deviance: 734.4 on 4997 degrees of freedom AIC: 740.4

Number of Fisher Scoring iterations: 8

Q6(b)

Boot function.

```
boot.fn <- function(data,index){
  fit.glm1.3 <- glm(default~income+balance, data=data[index, ], family = binomial)
  return(coef(fit.glm1.3))
}
boot.fn(Default, 1:10000)
```

(Intercept) income balance -1.154047e+01 2.080898e-05 5.647103e-03

Q6(c)

Standard error computation using bootstrap.

```
library(boot)
set.seed(1)
boot(Default,boot.fn,1000)
```

ORDINARY NONPARAMETRIC BOOTSTRAP

Call: boot(data = Default, statistic = boot.fn, R = 1000)

Bootstrap Statistics : original bias std. error t1* -1.154047e+01 -8.008379e-03 4.239273e-01 t2* 2.080898e-05 5.870933e-08 4.582525e-06 t3* 5.647103e-03 2.299970e-06 2.267955e-04

Q6(d)

Comments on (a) and (c).

The disparity in the estimated standard errors of the logistic regression and bootstrap may be attributed to the inadequacy of the fitted model in the logistic regression. Further, the bootstrap approach does not assume that the variablility only comes from the irreducible error, as compared to the logistic regression.

Question 8 - Cross-Validation (CV)

Q8(a)

Generate a simulated data.

```
set.seed(1)
y<-rnorm(100)
x<-rnorm(100)
y<-x-2*x^2+rnorm(100)
```

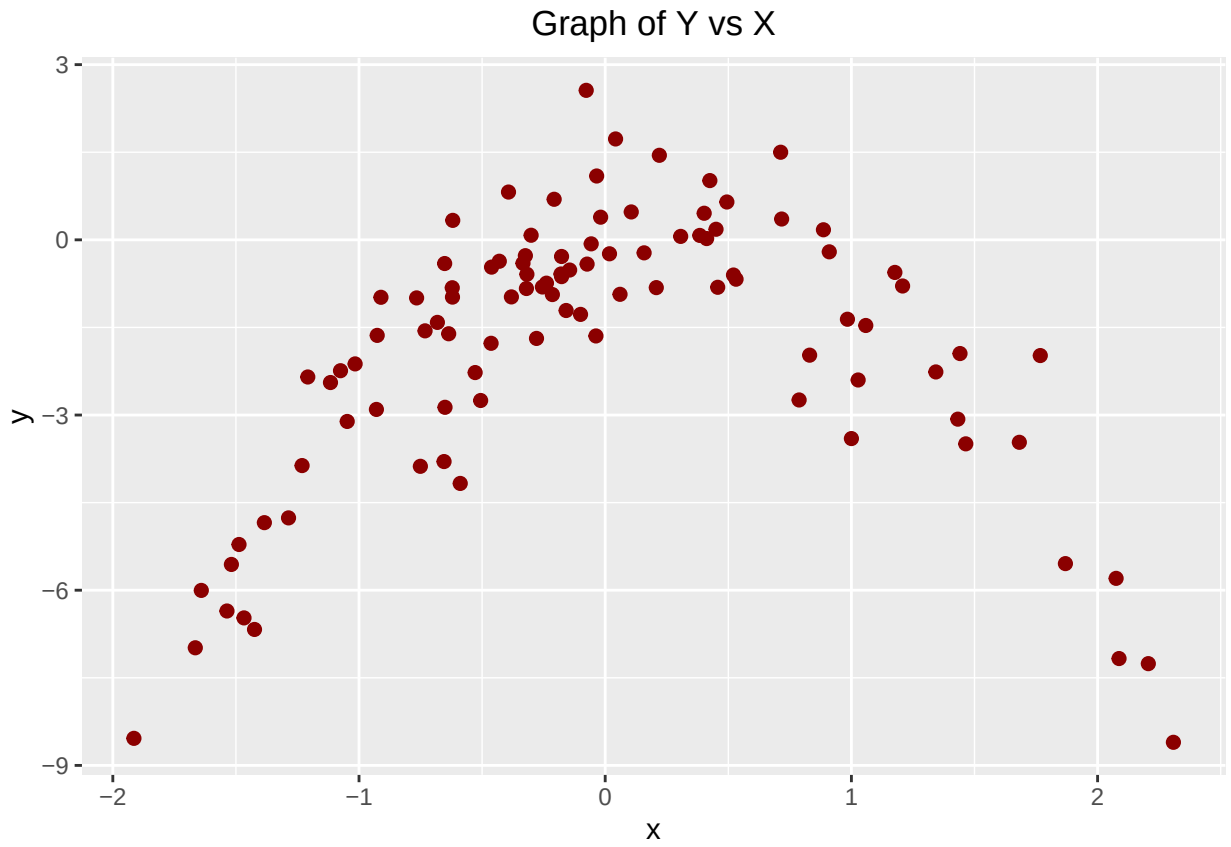
n = y = 100; p = x = 100

$$y = x - 2x^2 + \epsilon$$

Q8(b)

Scatterplot.

```
data<-data.frame(y,x)
library(ggplot2)
ggplot(data, aes(x=x, y=y))+
  geom_point(color = "darkred", size = 2)+
  ggtitle("Graph of Y vs X")+
  theme(plot.title = element_text(hjust = 0.5))
```



The graph obtained looks like an inverse quadratic graph. This suggests that the relationship between x and y is non-linear.

Q8(c)

LOOCV

```
set.seed(1)
library(boot)
cv.error=rep(0,4)
for (j in 1:4){
  glm.fit<-glm(y~poly(x, j), data = data)
  cv.error[j]<-cv.glm(data,glm.fit)$delta[1]
}
cv.error
```

[1] 5.890979 1.086596 1.102585 1.114772

Q8(d)

Repeat of (c) using different random seeds.

```
set.seed(2)
library(boot)
```

```
cv.error=rep(0,4)
for (j in 1:4){
  glm.fit<-glm(y~poly(x, j), data = data)
  cv.error[j]<-cv.glm(data,glm.fit)$delta[1]
}
cv.error
```

[1] 5.890979 1.086596 1.102585 1.114772

The LOOCV errors are the same. This is expected since there is no randomness in the tranining/validation data set splits.

Q8(e)

The quadratic model had the lowest LOOCV error. This is expected since y is a polynomial of the second order (i.e quadratic) that is dependent on x.

Q8(f)

Fitting w/ Least Squares.

```
set.seed(1)
for (j in 1:4){
  print(summary(glm(y~poly(x, j), data = data)))
}
```

Call: glm(formula = y ~ poly(x, j), data = data)

Deviance Residuals: Min 1Q Median 3Q Max
-7.3469 -0.9275 0.8028 1.5608 4.3974

Coefficients: Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.8277 0.2362 -7.737 9.18e-12 *** poly(x, j) 2.3164 2.3622 0.981 0.329
— Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 5.580018)

```
Null deviance: 552.21  on 99  degrees of freedom
```

Residual deviance: 546.84 on 98 degrees of freedom AIC: 459.69

Number of Fisher Scoring iterations: 2

Call: glm(formula = y ~ poly(x, j), data = data)

Deviance Residuals: Min 1Q Median 3Q Max
-2.89884 -0.53765 0.04135 0.61490 2.73607

Coefficients: Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.8277 0.1032 -17.704 <2e-16 ***poly(x, j)1 2.3164 1.0324 2.244 0.0271***
poly(x, j)2 -21.0586 1.0324 -20.399 <2e-16 * — Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 1.06575)

```
Null deviance: 552.21  on 99  degrees of freedom
```

Residual deviance: 103.38 on 97 degrees of freedom AIC: 295.11

Number of Fisher Scoring iterations: 2

Call: glm(formula = y ~ poly(x, j), data = data)

Deviance Residuals: Min 1Q Median 3Q Max
-2.87250 -0.53881 0.02862 0.59383 2.74350

Coefficients: Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.8277 0.1037 -17.621 <2e-16 ***poly(x, j)1 2.3164 1.0372 2.233 0.0279***
poly(x, j)2 -21.0586 1.0372 -20.302 <2e-16 * poly(x, j)3 -0.3048 1.0372 -0.294 0.7695
— Signif. codes: 0 ‘***’ 0.001 ’** 0.01 ’* 0.05 ‘.’ 0.1 ’ ’ 1

(Dispersion parameter for gaussian family taken to be 1.075883)

```
Null deviance: 552.21  on 99  degrees of freedom
```

Residual deviance: 103.28 on 96 degrees of freedom AIC: 297.02

Number of Fisher Scoring iterations: 2

Call: glm(formula = y ~ poly(x, j), data = data)

Deviance Residuals: Min 1Q Median 3Q Max
-2.8914 -0.5244 0.0749 0.5932 2.7796

Coefficients: Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.8277 0.1041 -17.549 <2e-16 ***poly(x, j)1 2.3164 1.0415 2.224 0.0285***
poly(x, j)2 -21.0586 1.0415 -20.220 <2e-16 * poly(x, j)3 -0.3048 1.0415 -0.293 0.7704
poly(x, j)4 -0.4926 1.0415 -0.473 0.6373
— Signif. codes: 0 ‘***’ 0.001 ’** 0.01 ’* 0.05 ‘.’ 0.1 ’ ’ 1

(Dispersion parameter for gaussian family taken to be 1.084654)

```
Null deviance: 552.21  on 99  degrees of freedom
```

Residual deviance: 103.04 on 95 degrees of freedom AIC: 298.78

Number of Fisher Scoring iterations: 2

The quadratic term in the quadratic, cubic, and quartic model is statistically significant, while the rest are not. This agrees with the conclusion drawn from the CV analysis that suggests that the quadratic model outperforms the rest.

Question 9 - Bootstrap

Q9(a)

Compute mean.

```
attach(Boston)
mu<-mean(medv)
mu
```

[1] 22.53281

Q9(b)

Standard error.

```
se<-sd(medv)/sqrt(length(medv))
se
```

[1] 0.4088611

Q9(c)

Standard error using bootstrap.

```
set.seed(1)
boot(medv,function(x,index){mean(x[index])},R<-1000)
```

ORDINARY NONPARAMETRIC BOOTSTRAP

Call: boot(data = medv, statistic = function(x, index) { mean(x[index]) }, R = R <- 1000)

Bootstrap Statistics : original bias std. error t1* 22.53281 0.008517589 0.4119374

The standard error is almost the same as the estimated standard error in (b).

Q9(d)

Confidence interval.

```
t.test(Boston$medv)
```

```
One Sample t-test
```

data: Boston$medv t = 55.111, df = 505, p-value < 2.2e-16 alternative hypothesis: true mean is not equal to 0 95 percent confidence interval: 21.72953 23.33608 sample estimates: mean of x 22.53281

```
CI.mu.hat<-c(mu-2*se,mu+2*se)
CI.mu.hat
```

[1] 21.71508 23.35053

The confidence interval obtained using the central limit theorem approach and t-test approach are almost the same.

Q9(e)

Compute median of medv.

```
mu.med<-median(medv)
mu.med
```

[1] 21.2

Q9(f)

Compute standard error using bootstrap.

```
set.seed(1)
boot(medv, function(x,index){median(x[index])},1000)
```

ORDINARY NONPARAMETRIC BOOTSTRAP

Call: boot(data = medv, statistic = function(x, index) { median(x[index]) }, R = 1000)

Bootstrap Statistics : original bias std. error t1* 21.2 -0.01615 0.3801002

The standard error of the median is approx. 0.378.

Q9(g)

Compute quantile.

```
mu0.1<-quantile(medv, .1)
mu0.1
```

10% 12.75

Q9(h)

Standard error of quantile using bootstrap.

```
set.seed(1)
boot(medv, function(x,index){quantile(medv[index],.1)},1000)
```

ORDINARY NONPARAMETRIC BOOTSTRAP

Call: boot(data = medv, statistic = function(x, index) { quantile(medv[index], 0.1) }, R = 1000)

Bootstrap Statistics : original bias std. error t1* 12.75 0.01005 0.505056

The estimated standard error of the 10th percentile of medv in Boston suburbs is 0.477.