Assignment 3 :glm and MSE practice

Andrew Abisha Hu, Elaine Su, Eric Xiong, Tianye Wang

9/5/2018

A marketing research company conduct a survey to investigate consumers' willingness to try anew energy drink. Download energy.csv on Canvas and answer the following questions. Here is the description of variables:

```
sex: sex of the respondent (0 = female, 1 = male)
age: age group of the respondent (10-19, 20-29, 30-39)
exercise: self-reported exercise frequency (0 = less than twice/week, 1 = at least
twice/week)
innovation: a composite index of the respondent's tendency to try new things
user: energy drink user (0 = never drink any energy drink, 1 = others)
try: try the new energy drink after reading its advertising flyer (0 = no, 1 = yes)
train: training data set (if TURE)
cv: indicators for 5-fold cross validation (data sets 0 - 4)
```

1. Use only the training data set to run a Probit model. Predict try by all other variables except for train and cv. (provide R codes, 0.5%)

```r
library(car)
```

```
## Loading required package: carData
```

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
setwd("/Users/andrewhu/Documents/GitHub/Machine-Learning/Class Practice/Assignment 3-glm")
energy <- read.csv("energy.csv")
baseball <- read.csv("baseball.csv")

fit = glm( try~sex+age + exercise+ innovation +user, data= energy, family=binomial(link="probit"), subs
```

2. Compute p-value for the overall model significant. (provide R codes, 0.3%)

```r
1- pchisq(fit$null.deviance- fit$deviance, 6)
```

```
## [1] 0
```

3. Based on the p-value, what is your conclusion? (0.2%)

   `We can reject the null hypothesis → we conclude that at least one variable has a significant effec`

4. Now use the test data set (i.e., non-training data set) to compute the classification rate. You consider that a respondent will try the new energy drink if the predicted probability is at least 0.5. (provide R codes, 0.5%)

```r
tab= table(energy$try[energy$train==FALSE],predict(fit,energy[energy$train==FALSE,], type="response")>.
sum(diag(tab))/ sum(tab)
```

```
## [1] 0.6877682
```

5. You suspect that males who exercise at least twice per week may be more likely to try the new product than other males. How will you expand the Probit model to test your hypothesis? (provide R codes, 0.5%)

```r
fit2 = glm(try~sex +age+ exercise + innovation + user +sex* exercise, data=energy, family=binomial(link=
```

6. What is the 99% confidence interval of the coefficient of the new variable you added? Based on the confidence interval, what is your conclusion? (provide R codes, 0.5%)

```r
confint(fit2, level=.99)
```

```
## Waiting for profiling to be done...

##                    0.5 %     99.5 %
## (Intercept)  -0.1625096  0.3193868
## sex          -0.2307351  0.2236925
## age20~29     -0.2168893  0.1668753
## age30~39     -0.6448937 -0.2704766
## exercise      0.3272376  0.7316224
## innovation   -0.2120895  0.3127996
## user          0.0928167  0.4757708
## sex:exercise  0.1650906  0.7871330
```

```
[0.1650906, 0.7871330] → does not include 0, so we can conclude that at the 0.01 level,
there is a significant difference between males who exercise at least twice per week and
males who exercise less than twice a week.
```

7. Find AIC of the two models. Based on AIC, would you prefer the original model or the expanded one? (0.5%)

```
AIC of original model: 2451.3
AIC of expanded model: 2437.7
The expanded model is better than the original model as it has a lower AIC. Also, the
difference of their AIC is larger than 10, which is a strong evidence in favor of the lower
AIC model (the expanded model)
```

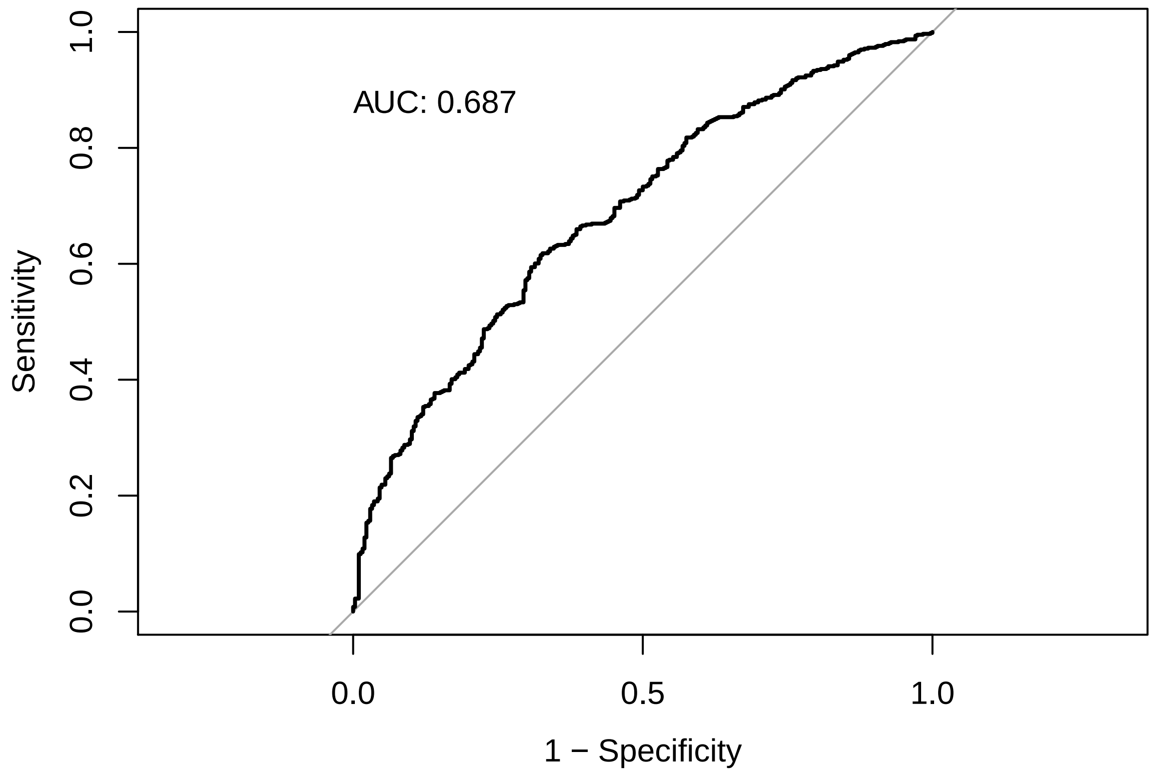
8. Now use the test data set (i.e., non-training data set) and compute the classification rate of the expanded model. You consider that a respondent will try the new energy drink if the predicted probability is at least 0.5. (provide R codes, 0.5%)

```r
tab2 = table(energy[energy$train==FALSE,]$try, predict(fit2, energy[energy$train==FALSE,], type= "respon
sum(diag(tab2))/sum(tab2)
```

```
## [1] 0.6920601
```

9. Use the test data set to create ROC plots and compute AUC for both models. (provide R codes, 0.5%)

```r
plot.roc(energy$try[energy$train==F], predict(fit,energy[energy$train==F,],type="response"), data=energy
```




```
#plot.roc(energy[energy$train==F,]$try, predict(fit2,energy[energy$train==F,],type="response"), data=en
```

10. Based on AUC, would you prefer the original model or the expanded one? (0.5%)

```
We can see that the original model has a slightly higher AUC, which is preferred over the
expanded model as it indicates higher accuracy
```

11. Do you reach the same conclusion when using AIC and AUC? Discuss what might happen. (0.5%)

```
No. The expanded model has lower AIC in the training data set, which means that the
expanded model has higher predictive accuracy in the training data set than the original
model. However, the reason is that the expanded model may overfit the training data set
and only capture the idiosyncrasies of this particular data set but may not predict well for
other data sets. This is why when we use test data set to do out-of-sample prediction, the
expanded model has lower AUC than the original one. Using in-sample AIC can be too
optimistic in evaluating prediction accuracy.
```

12. Use cv to perform 5-fold cross validation for the original model in Question 1. Compute the AUC for each fold. What is the average AUC over the 5 folds? (provide R codes, 0.5%) Hint: you can use auc(true responses, predicted probabilities) in the pROCpackage to compute AUC.

```
yhat = rep(NA, nrow(energy))

for (i in 0:4) {
        fit3 = glm(try~sex + age + exercise + innovation + user, data=energy, family=binomial(link="log
        yhat[energy$cv==i] = predict(fit3, energy[energy$cv==i,])

}
mean((energy$try-yhat)^2)
```

```
## [1] 0.5458279
```

```r
auc(energy$try, yhat)
```

```
## Area under the curve: 0.6898
```

13. Repeat question (12) for the expanded model. What is the average AUC over the 5 folds? (provide R codes, 0.5%)

```r
yhat2 = rep(NA, nrow(energy))

for (i in 0:4) {
        fit4 = glm(try~sex + age + exercise + innovation + user + sex*exercise, data=energy, family=bino
        yhat2[energy$cv==i] = predict(fit4, energy[energy$cv==i,])

}
mean((energy$try-yhat2)^2)
```

```
## [1] 0.5819057
```

```r
auc(energy$try, yhat2)
```

```
## Area under the curve: 0.688
```

14. Based on the average AUC you obtained from question (12) and (13), comment on which model is preferred. (0.5%) Hint: you may want to consider the principle of parsimony.

```
We want to build a parsimonious model with good predictive accuracy. According to the
average AUC obtained, the original model is preferred as it has a slightly higher AUC,
which means that it has a stronger predictive power. If a variable doesn't contribute much
to prediction we don't necessarily need to include it
```

Download baseball.csv and answer the following questions. Here is some background information of the data: "The baseball salary data set is available through the Journal of StatisticsEducation (JSE) data archive. The data set contains salary information for 337 Major League Baseball (MLB) players who are not pitchers and played at least one game during both the 1991 and 1992 seasons. The purpose of the study is to determine whether a baseball player's salary is a reflection of his offensive performance. For each player, the salary from the 1992 season along with 12 offensive statistics from the 1991 season were collected. In addition to these variables,there are 4 indicator variables which identify free agency and eligibility for arbitration."(https://www4.stat.ncsu.edu/~boos/var.select/baseball.html)

```
y: Salary (in thousands of dollars)
x1: Batting average
x2: On-base percentage
x3: Number of runs
x4: Number of hits
x5: Number of doubles
x6: Number of triples
x7: Number of home runs
x8: Number of runs batted in
x9: Number of walks
x10: Number of strike-outs
x11: Number of stolen bases
x12: Number of errors
x13: Indicator of "free agency eligibility"
x14: Indicator of "free agent in 1991/2"
x15: Indicator of "arbitration eligibility"
x16: Indicator of "arbitration in 1991/2"
```

15. Full model: Use only the first 200 observations (i.e., the training data set) to regress log(y)on all predictors. (provide R codes, 0.5%)

```
train= baseball[c(1:200),]
test= baseball[-c(1:200),]
fit5 = lm(log(y)~.,data=train)
```

16. Define the square error of an observation as (log(y) – log(y)_hat)^2. Use the full model to compute the average square error of the test set (i.e., the last 137 observations). (provide R codes, 0.5%)

```
mean((log(baseball$y[-c(1:200)])-predict(fit5,test))^2)
```

```
## [1] 0.3452043
```

17. Use the training data set and perform a backward selection for the model. (Provide R codes, 0.5%)

```
fit6 = lm(log(y)~.,train)
step(fit6)
```

```
## Start:  AIC=-249.31
## log(y) ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 +
##     x12 + x13 + x14 + x15 + x16
##
##        Df Sum of Sq     RSS      AIC
## - x1    1     0.014  48.525 -251.249
## - x2    1     0.082  48.592 -250.970
## - x7    1     0.110  48.620 -250.855
## - x9    1     0.163  48.674 -250.636
## - x3    1     0.169  48.679 -250.614
## - x4    1     0.170  48.680 -250.609
## - x5    1     0.225  48.735 -250.384
## - x6    1     0.248  48.758 -250.289
## - x11   1     0.257  48.767 -250.252
## - x10   1     0.310  48.821 -250.033
## - x16   1     0.395  48.906 -249.685
## - x12   1     0.455  48.965 -249.443
## <none>               48.510 -249.309
## - x14   1     0.642  49.152 -248.679
## - x8    1     1.553  50.064 -245.004
## - x15   1    38.592  87.102 -134.247
## - x13   1    62.634 111.144  -85.497
##
## Step:  AIC=-251.25
## log(y) ~ x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10 + x11 +
##     x12 + x13 + x14 + x15 + x16
##
##        Df Sum of Sq     RSS      AIC
## - x7    1     0.115  48.640 -252.774
## - x2    1     0.147  48.672 -252.644
## - x9    1     0.170  48.695 -252.548
## - x3    1     0.173  48.698 -252.537
## - x5    1     0.231  48.756 -252.297
## - x4    1     0.233  48.757 -252.292
## - x6    1     0.251  48.776 -252.215
## - x11   1     0.252  48.777 -252.213
## - x10   1     0.327  48.852 -251.905
## - x16   1     0.403  48.928 -251.595
```

```
## - x12   1     0.457  48.982 -251.375
## <none>              48.525 -251.249
## - x14   1     0.645  49.170 -250.607
## - x8    1     1.592  50.116 -246.795
## - x15   1    38.737  87.261 -135.882
## - x13   1    63.141 111.666  -86.561
##
## Step:  AIC=-252.77
## log(y) ~ x2 + x3 + x4 + x5 + x6 + x8 + x9 + x10 + x11 + x12 +
##     x13 + x14 + x15 + x16
##
##        Df Sum of Sq     RSS      AIC
## - x3    1     0.082  48.722 -254.437
## - x2    1     0.187  48.827 -254.005
## - x6    1     0.189  48.829 -254.000
## - x5    1     0.227  48.867 -253.843
## - x11   1     0.329  48.969 -253.425
## - x16   1     0.356  48.996 -253.316
## - x9    1     0.371  49.011 -253.255
## - x12   1     0.417  49.057 -253.065
## <none>              48.640 -252.774
## - x14   1     0.565  49.205 -252.463
## - x10   1     0.641  49.281 -252.157
## - x4    1     0.650  49.290 -252.121
## - x8    1     2.484  51.124 -244.812
## - x15   1    38.842  87.482 -137.376
## - x13   1    64.798 113.438  -85.412
##
## Step:  AIC=-254.44
## log(y) ~ x2 + x4 + x5 + x6 + x8 + x9 + x10 + x11 + x12 + x13 +
##     x14 + x15 + x16
##
##        Df Sum of Sq     RSS      AIC
## - x6    1     0.161  48.883 -255.779
## - x2    1     0.190  48.911 -255.661
## - x5    1     0.202  48.924 -255.610
## - x16   1     0.334  49.056 -255.069
## - x12   1     0.385  49.107 -254.865
## <none>              48.722 -254.437
## - x14   1     0.623  49.344 -253.898
## - x11   1     0.642  49.364 -253.819
## - x10   1     0.643  49.365 -253.814
## - x9    1     0.773  49.495 -253.289
## - x4    1     1.135  49.857 -251.832
## - x8    1     2.940  51.661 -244.721
## - x15   1    38.760  87.482 -139.376
## - x13   1    65.941 114.663  -85.264
##
## Step:  AIC=-255.78
## log(y) ~ x2 + x4 + x5 + x8 + x9 + x10 + x11 + x12 + x13 + x14 +
##     x15 + x16
##
##        Df Sum of Sq     RSS      AIC
## - x2    1     0.193  49.075 -256.993
```

```
## - x5    1     0.200  49.083 -256.961
## - x16   1     0.333  49.215 -256.423
## - x12   1     0.345  49.228 -256.371
## <none>                48.883 -255.779
## - x11   1     0.510  49.393 -255.702
## - x14   1     0.615  49.498 -255.278
## - x10   1     0.712  49.594 -254.889
## - x9    1     0.949  49.831 -253.934
## - x4    1     0.979  49.862 -253.813
## - x8    1     3.116  51.999 -245.418
## - x15   1    38.600  87.482 -141.376
## - x13   1    65.797 114.679  -87.236
##
## Step:  AIC=-256.99
## log(y) ~ x4 + x5 + x8 + x9 + x10 + x11 + x12 + x13 + x14 + x15 +
##     x16
##
##        Df Sum of Sq     RSS     AIC
## - x5    1     0.189  49.264 -258.23
## - x12   1     0.337  49.412 -257.63
## - x16   1     0.358  49.433 -257.54
## <none>                49.075 -256.99
## - x11   1     0.508  49.583 -256.93
## - x10   1     0.537  49.612 -256.82
## - x14   1     0.580  49.655 -256.64
## - x9    1     0.758  49.833 -255.93
## - x4    1     0.890  49.965 -255.40
## - x8    1     3.061  52.136 -246.89
## - x15   1    39.864  88.939 -140.07
## - x13   1    66.199 115.275  -88.20
##
## Step:  AIC=-258.23
## log(y) ~ x4 + x8 + x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16
##
##        Df Sum of Sq     RSS      AIC
## - x12   1     0.343  49.606 -258.840
## - x16   1     0.352  49.616 -258.800
## - x10   1     0.466  49.730 -258.343
## <none>                49.264 -258.225
## - x14   1     0.504  49.768 -258.191
## - x11   1     0.611  49.874 -257.762
## - x4    1     0.745  50.009 -257.224
## - x9    1     0.797  50.061 -257.014
## - x8    1     2.875  52.139 -248.882
## - x15   1    40.107  89.371 -141.104
## - x13   1    67.184 116.447  -88.176
##
## Step:  AIC=-258.84
## log(y) ~ x4 + x8 + x9 + x10 + x11 + x13 + x14 + x15 + x16
##
##        Df Sum of Sq     RSS      AIC
## - x16   1     0.384  49.990 -259.299
## <none>                49.606 -258.840
## - x4    1     0.521  50.128 -258.748
```

```
## - x10   1     0.607  50.214 -258.407
## - x14   1     0.657  50.263 -258.209
## - x11   1     0.713  50.319 -257.987
## - x9    1     0.798  50.404 -257.648
## - x8    1     3.153  52.760 -248.514
## - x15   1    39.819  89.425 -142.983
## - x13   1    69.067 118.674  -86.388
##
## Step:  AIC=-259.3
## log(y) ~ x4 + x8 + x9 + x10 + x11 + x13 + x14 + x15
##
##        Df Sum of Sq     RSS      AIC
## - x4    1     0.475  50.465 -259.406
## <none>               49.990 -259.299
## - x10   1     0.633  50.623 -258.783
## - x14   1     0.656  50.646 -258.692
## - x11   1     0.776  50.766 -258.219
## - x9    1     0.865  50.855 -257.868
## - x8    1     3.199  53.189 -248.894
## - x15   1    41.672  91.662 -140.042
## - x13   1    69.008 118.998  -87.843
##
## Step:  AIC=-259.41
## log(y) ~ x8 + x9 + x10 + x11 + x13 + x14 + x15
##
##        Df Sum of Sq     RSS      AIC
## <none>               50.465 -259.406
## - x14   1     0.694  51.160 -258.673
## - x10   1     0.731  51.196 -258.532
## - x9    1     1.292  51.757 -256.351
## - x11   1     1.526  51.991 -255.450
## - x8    1     8.289  58.754 -230.991
## - x15   1    46.986  97.452 -129.792
## - x13   1    71.863 122.328  -84.322
```

```
##
## Call:
## lm(formula = log(y) ~ x8 + x9 + x10 + x11 + x13 + x14 + x15,
##     data = train)
##
## Coefficients:
## (Intercept)           x8           x9          x10          x11
##    5.030624     0.012405     0.005409    -0.002908     0.008266
##         x13          x14          x15
##    1.667756    -0.219651     1.386866
```

18. What is the estimated regression equation after the backward selection? (0.5%)

```r
summary(lm(formula= log(y)~ x8 +x9+ x10+x11+x13+ x14+ x15 ,train))
```

```
##
## Call:
## lm(formula = log(y) ~ x8 + x9 + x10 + x11 + x13 + x14 + x15,
##     data = train)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -1.65580 -0.34291 -0.06054  0.34822  1.34186
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.030624   0.076620  65.656  < 2e-16 ***
## x8           0.012405   0.002209   5.616 6.79e-08 ***
## x9           0.005409   0.002440   2.217   0.0278 *
## x10         -0.002908   0.001744  -1.667   0.0971 .
## x11          0.008266   0.003431   2.409   0.0169 *
## x13          1.667756   0.100862  16.535  < 2e-16 ***
## x14         -0.219651   0.135136  -1.625   0.1057
## x15          1.386866   0.103728  13.370  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5127 on 192 degrees of freedom
## Multiple R-squared:  0.8087, Adjusted R-squared:  0.8017
## F-statistic: 115.9 on 7 and 192 DF,  p-value: < 2.2e-16
```

19. Use the backward selection regression to compute the average square error of the test set. (provide R codes, 0.5%)

```r
fit7 = lm(formula= log(y)~ x8+ x9+ x10+x11+x13+ x14+ x15, train)
mean((log(baseball$y[-c(1:200)])-predict(fit7,test))^2)
```

```
## [1] 0.3510936
```

20. Use the training data set and perform a forward selection, starting with only the intercept and considering all predictors. (Provide R codes, 0.5%)

```r
fit8 = lm(log(y)~1, train)
step(fit8, scope=~x1+x2+x3+x4+x5+x6+x7+x8+ x9+ x10+x11+x12+x13+ x14+ x15+x16, direction="forward")
```

```
## Start:  AIC=57.34
## log(y) ~ 1
##
##        Df Sum of Sq    RSS     AIC
## + x3    1   117.536 146.22 -58.640
## + x4    1   116.236 147.52 -56.869
## + x8    1   106.348 157.41 -43.894
## + x13   1   102.244 161.51 -38.747
## + x9    1   101.505 162.25 -37.834
## + x5    1    84.519 179.24 -17.921
## + x7    1    69.302 194.46  -1.623
## + x10   1    53.050 210.71  14.430
## + x2    1    29.395 234.36  35.710
## + x6    1    27.218 236.54  37.559
## + x12   1    17.601 246.16  45.529
## + x15   1    16.793 246.96  46.185
## + x11   1    16.286 247.47  46.595
## + x1    1    15.003 248.75  47.629
## + x14   1     7.378 256.38  53.668
## <none>              263.76  57.342
## + x16   1     0.815 262.94  58.723
##
```

```
## Step:  AIC=-58.64
## log(y) ~ x3
##
##         Df Sum of Sq     RSS      AIC
## + x13   1    46.370  99.851 -132.927
## + x14   1     7.645 138.576  -67.380
## + x15   1     6.974 139.247  -66.414
## + x8    1     4.906 141.315  -63.466
## + x4    1     3.653 142.568  -61.700
## + x11   1     3.065 143.156  -60.877
## + x9    1     2.667 143.554  -60.322
## <none>              146.221  -58.640
## + x7    1     1.427 144.793  -58.602
## + x6    1     0.454 145.767  -57.261
## + x12   1     0.315 145.905  -57.072
## + x1    1     0.240 145.981  -56.968
## + x10   1     0.177 146.044  -56.882
## + x5    1     0.101 146.120  -56.778
## + x16   1     0.094 146.127  -56.768
## + x2    1     0.001 146.220  -56.642
##
## Step:  AIC=-132.93
## log(y) ~ x3 + x13
##
##         Df Sum of Sq    RSS     AIC
## + x15   1    45.359 54.492 -252.05
## + x4    1     4.792 95.059 -140.76
## + x8    1     3.677 96.174 -138.43
## + x16   1     2.482 97.369 -135.96
## + x5    1     1.277 98.574 -133.50
## <none>              99.851 -132.93
## + x11   1     0.875 98.976 -132.69
## + x7    1     0.429 99.422 -131.79
## + x2    1     0.322 99.529 -131.57
## + x12   1     0.255 99.597 -131.44
## + x14   1     0.089 99.762 -131.11
## + x9    1     0.085 99.766 -131.10
## + x1    1     0.054 99.798 -131.03
## + x10   1     0.043 99.809 -131.01
## + x6    1     0.004 99.847 -130.94
##
## Step:  AIC=-252.05
## log(y) ~ x3 + x13 + x15
##
##         Df Sum of Sq    RSS     AIC
## + x8    1   2.62231 51.870 -259.92
## + x7    1   0.75111 53.741 -252.83
## + x12   1   0.65891 53.833 -252.49
## + x4    1   0.60413 53.888 -252.28
## <none>              54.492 -252.05
## + x16   1   0.53412 53.958 -252.02
## + x14   1   0.48393 54.008 -251.84
## + x6    1   0.32741 54.165 -251.26
## + x9    1   0.31182 54.181 -251.20
```

```
## + x5     1   0.22187 54.270 -250.87
## + x11    1   0.09557 54.397 -250.40
## + x10    1   0.02378 54.469 -250.14
## + x1     1   0.00408 54.488 -250.07
## + x2     1   0.00119 54.491 -250.06
##
## Step:  AIC=-259.92
## log(y) ~ x3 + x13 + x15 + x8
##
##        Df Sum of Sq    RSS     AIC
## + x12   1   0.70358 51.166 -260.65
## + x16   1   0.52764 51.342 -259.96
## <none>              51.870 -259.92
## + x14   1   0.48116 51.389 -259.78
## + x10   1   0.37604 51.494 -259.37
## + x7    1   0.32403 51.546 -259.17
## + x11   1   0.23635 51.634 -258.83
## + x9    1   0.18505 51.685 -258.63
## + x6    1   0.09277 51.777 -258.27
## + x4    1   0.08892 51.781 -258.26
## + x5    1   0.03676 51.833 -258.06
## + x1    1   0.00966 51.860 -257.95
## + x2    1   0.00850 51.862 -257.95
##
## Step:  AIC=-260.65
## log(y) ~ x3 + x13 + x15 + x8 + x12
##
##        Df Sum of Sq    RSS     AIC
## <none>              51.166 -260.65
## + x16   1   0.46822 50.698 -260.49
## + x7    1   0.45447 50.712 -260.43
## + x14   1   0.32760 50.839 -259.93
## + x10   1   0.28339 50.883 -259.76
## + x4    1   0.20440 50.962 -259.45
## + x11   1   0.15525 51.011 -259.25
## + x9    1   0.15298 51.013 -259.25
## + x6    1   0.12282 51.044 -259.13
## + x5    1   0.01576 51.151 -258.71
## + x1    1   0.00890 51.158 -258.68
## + x2    1   0.00168 51.165 -258.65

##
## Call:
## lm(formula = log(y) ~ x3 + x13 + x15 + x8 + x12, data = train)
##
## Coefficients:
## (Intercept)           x3          x13          x15           x8
##    4.985781     0.009736     1.595164     1.388692     0.007331
##         x12
##   -0.011235
```

21. What is the estimated regression equation of the forward selection? (0.5%)

```
summary(lm(formula=log(y)~ x3+x13+x15+x8+x12,train))
```

```
##
```

```
## Call:
## lm(formula = log(y) ~ x3 + x13 + x15 + x8 + x12, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6218 -0.3288 -0.0440  0.3549  1.2685
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.985781   0.072967  68.329  < 2e-16 ***
## x3           0.009736   0.002511   3.877 0.000145 ***
## x13          1.595164   0.090593  17.608  < 2e-16 ***
## x15          1.388692   0.106553  13.033  < 2e-16 ***
## x8           0.007331   0.002305   3.180 0.001715 **
## x12         -0.011235   0.006879  -1.633 0.104029
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5136 on 194 degrees of freedom
## Multiple R-squared:  0.806,  Adjusted R-squared:  0.801
## F-statistic: 161.2 on 5 and 194 DF,  p-value: < 2.2e-16
```

22. Use the forward selection regression to compute the average square error of the test set. (provide R codes, 0.5%)

```r
fit9 = lm(log(y)~x3 + x13+ x15 + x8 + x12, train)
mean((log(baseball$y[-c(1:200)])-predict(fit9,test))^2)
```

```
## [1] 0.3714355
```

23. Does every variable selected by iterative model selection methods have significant effects on the dependent variable? Run a model with only selected variables to show your answer. (0.5%)

```
Not every variable selected by iterative model selection methods have significant effects on the
dependent variable
```

24. Why do you think iterative model selection methods will or will not pick only variables with significant effects? (0.5%)

```
Iterate model selection methods selects the model based on AIC values, not p-values. The goal is
to find the model with the smallest AIC by removing or adding variables in the scope
```

25. Use the training data set to run a ridge regression with all predictors. Set the random seed as set.seed(12345) and use glmnet with 5-fold cross validation to find the optimal lambda with [0,1] (i.e., seq(0,1,0.0001)). What is the optimal lambda value? (provide R codes, 0.5%)

```r
library(glmnet)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-16

##
## Attaching package: 'glmnet'

## The following object is masked from 'package:pROC':
##
##     auc
```

```r
lam= seq(0,1,0.0001)
set.seed(12345)
fit10= cv.glmnet(as.matrix(baseball[1:200, 2:17]), log(baseball$y[1:200]), alpha=0,nfolds=5, lambda=lam
fit10$lambda.min
```

```
## [1] 0.0363
```

26. What is the estimated regression equation of the ridge model with the optimal lambda value? (0.5%)

```r
summary(fit10)
```

```
##            Length Class  Mode
## lambda     10001  -none- numeric
## cvm        10001  -none- numeric
## cvsd       10001  -none- numeric
## cvup       10001  -none- numeric
## cvlo       10001  -none- numeric
## nzero      10001  -none- numeric
## name           1  -none- character
## glmnet.fit    12  elnet  list
## lambda.min     1  -none- numeric
## lambda.1se     1  -none- numeric
```

27. Use the ridge regression with the optimal lambda value to compute the average square error of the test set (provide R codes, 0.5%).

```r
newx = model.matrix(log(y)~. , test)[,2:17]
mean((log(baseball$y[-c(1:200)])-predict(fit10, newx, s="lambda.min"))^2)
```

```
## [1] 0.3437116
```

28. Use the training data set to run a lasso regression with all predictors. Set the random seed as set.seed(12345) and use glmnet with 5-fold cross validation to find the optimal lambda with [0,1] (i.e., seq(0,1,0.0001)). What is the optimal lambda value? (provide R codes, 0.5%)

```r
set.seed(12345)
fit11= cv.glmnet(as.matrix(baseball[1:200,2:17]), log(baseball$y[1:200]), alpha=1, nfolds=5, lambda=lam
fit11$lambda.min
```

```
## [1] 0.0404
```

29. Use the training data and the optimal lambda value to run a lasso regression. What is the estimated regression equation of the lasso model? (0.5%)

```r
summary(fit11)
```

```
##            Length Class  Mode
## lambda     10001  -none- numeric
## cvm        10001  -none- numeric
## cvsd       10001  -none- numeric
## cvup       10001  -none- numeric
## cvlo       10001  -none- numeric
## nzero      10001  -none- numeric
## name           1  -none- character
## glmnet.fit    12  elnet  list
## lambda.min     1  -none- numeric
## lambda.1se     1  -none- numeric
```

30.Use the lasso regression with the optimal lambda value to compute the average square error of the test set (provide R codes, 0.5%)

```
newx2 = model.matrix(log(y)~. ,test)[,2:17]
mean((log(baseball$y[-c(1:200)])-predict(fit11, newx2, s= "lambda.min"))^2)
```

```
## [1] 0.374002
```

31. Comment on the performance of the full, backward, forward, ridge, and lasso models in predicting the test data set. (0.5%)

```
Full model square error: 0.3452043
Backward model square error: 0.3510936
Forward model square error: 0.3714355
Ridge model square error: 0.3437116
Lasso model square error: 0.374002
```

```
Ridge > Full > Backward > Forward > Lasso
Among all methods, the Ridge model has the best performance on predicting the test data set
```