Data 3 HW 2

Sean Duan

9/23/2020

1.

A

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

```
## starting httpd help server ... done
```

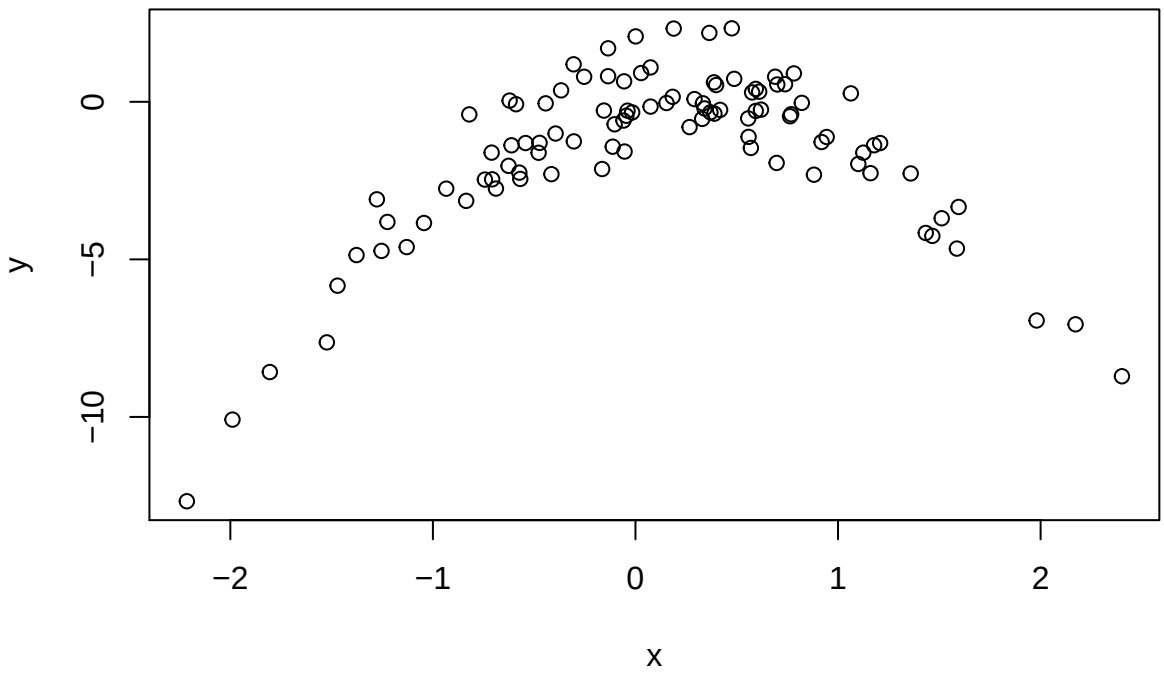
For this formula n is 100, and p is 2.

$$x \sim N(0,1)$$

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

B

```
plot(x,y)
```



It seems like nonlinear relationship between x and y, given that the functional form of the data follows a parabola.

C

```
set.seed(3)

#loocv
data_1<-as.data.frame(cbind(x,y))


cv.error=rep(0,4)
for (i in 1:4){
  glm.fit=glm(y~poly(x ,i),data=data_1)
  cv.error[i]=cv.glm(data_1 ,glm.fit)$delta [1]
}
cv.error
```

```
## [1] 7.2881616 0.9374236 0.9566218 0.9539049
```

Each element in the list above is analogous to the error for the 1st, 2nd, 3rd, and 4th order polynomials respectively.

D

```
set.seed(4)
```

```r
#loocv
data_1<-as.data.frame(cbind(x,y))


cv.error=rep(0,4)
for (i in 1:4){
  glm.fit=glm(y~poly(x ,i),data=data_1)
  cv.error[i]=cv.glm(data_1 ,glm.fit)$delta [1]
}
cv.error
```

```
## [1] 7.2881616 0.9374236 0.9566218 0.9539049
```

We see no difference with a different random seed, because no sampling is used in LOOCV. The entire data set is fit observation by observation for every element.

E

The model with the smallest LOOCV error is our second model, the one with the 2nd order polynomial on our predictor. This is exactly what I expected, as our plot of the effects of x on y had a parabolic arc in it's form.

F

```r
for (i in 1:4){
  print(summary(glm(y~poly(x ,i),data=data_1)))
  }
```

```
##
## Call:
## glm(formula = y ~ poly(x, i), data = data_1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -9.5161  -0.6800   0.6812   1.5491   3.8183
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.550      0.260  -5.961 3.95e-08 ***
## poly(x, i)     6.189      2.600   2.380   0.0192 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 6.760719)
##
##     Null deviance: 700.85  on 99  degrees of freedom
## Residual deviance: 662.55  on 98  degrees of freedom
## AIC: 478.88
##
## Number of Fisher Scoring iterations: 2
##
##
## Call:
## glm(formula = y ~ poly(x, i), data = data_1)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9650  -0.6254  -0.1288   0.5803   2.2700
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.5500     0.0958  -16.18  < 2e-16 ***
## poly(x, i)1   6.1888     0.9580    6.46 4.18e-09 ***
## poly(x, i)2 -23.9483     0.9580  -25.00  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9178258)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  89.029  on 97  degrees of freedom
## AIC: 280.17
##
## Number of Fisher Scoring iterations: 2
##
##
## Call:
## glm(formula = y ~ poly(x, i), data = data_1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9765  -0.6302  -0.1227   0.5545   2.2843
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09626 -16.102  < 2e-16 ***
## poly(x, i)1   6.18883    0.96263   6.429 4.97e-09 ***
## poly(x, i)2 -23.94830    0.96263 -24.878  < 2e-16 ***
## poly(x, i)3   0.26411    0.96263   0.274    0.784
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9266599)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  88.959  on 96  degrees of freedom
## AIC: 282.09
##
## Number of Fisher Scoring iterations: 2
##
##
## Call:
## glm(formula = y ~ poly(x, i), data = data_1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0550  -0.6212  -0.1567   0.5952   2.2267
##
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09591 -16.162  < 2e-16 ***
## poly(x, i)1   6.18883    0.95905   6.453 4.59e-09 ***
## poly(x, i)2 -23.94830    0.95905 -24.971  < 2e-16 ***
## poly(x, i)3   0.26411    0.95905   0.275    0.784
## poly(x, i)4   1.25710    0.95905   1.311    0.193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9197797)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  87.379  on 95  degrees of freedom
## AIC: 282.3
##
## Number of Fisher Scoring iterations: 2
```

We see that the exponential term up to 2 is significant in all our models, as our previous answer would lead us to expect due to the form of our graph that we diagrammed earlier. For our models, the polynomial 3 and 4 models coefficients past the 2nd exponent are not significant, as a parabolic relationship would suggest.

2.

A

```
mean(Boston$medv)
```

```
## [1] 22.53281
```

B

```
(sd(Boston$medv))/sqrt(length(Boston$medv))
```

```
## [1] 0.4088611
```

Given the value of our mean, at approximately 22.5, having a standard error of .4 seems proportionally very small. Thus, we can conclude that we are fairly confident in our estimate.

C

```
boot_fn<-function(data, index){
  mean(data[index])
}
boot_result<-boot(data = Boston$medv, statistic = boot_fn, R=1000)
boot_result
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot_fn, R = 1000)
##
##
```

```
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 22.53281 -0.01170988   0.4188752
```

The standard error of our estimate of the mean is very slightly larger using the bootstrap. However, given the size of our mean value, and the proportional change as compared to our previous estimate of the standard error, we can conclude that there isn't a significant difference if we use the bootstrap to estimate our standard error.

D

```
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
```

```
mean(Boston$medv)-2*0.4012719
```

```
## [1] 21.73026
```

```
mean(Boston$medv)+2*0.4012719
```

```
## [1] 23.33535
```

Our results that we obtained using the t.test function are extremely similar to our 95% confidence interval we calculated using the boostrap. This provides further confidence that our bootstrap calculation has no significant difference from our other method of estimation.

E

```
median(Boston$medv)
```

```
## [1] 21.2
```

F

```
boot_fn2<-function(data, index){
  median(data[index])
}
boot_result2<-boot(data = Boston$medv, statistic = boot_fn2, R=1000)
boot_result2
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot_fn2, R = 1000)
```

```
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     21.2 -0.0252   0.3770837
```

The size of the standard error is similar to that which we have found earlier for the mean. This makes sense as our median estimate is less sensitive to outlier information, however, our data doesn't seem to have a great deal of outliers. Thus, it makes sense that our mean and median estimates would be similar

G

```
quantile(Boston$medv, probs = .1)
```

```
##   10%
## 12.75
```

H

```
boot_fn3<-function(data, index){
  quantile(data[index], probs = .1)
}
boot_result3<-boot(data = Boston$medv, statistic = boot_fn3, R=1000)
boot_result3
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot_fn3, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75  0.0508   0.4847518
```

Our estimation of standard error seems similar for the 10th percentile of our value as compared to our estimation for the standard errors of of mean and median. This lends us some confidence that our data-set is relatively uniform throughout it's entire range of recorded values.

3.

```
set.seed(1)

mods=list(1,2,3,4,c(1,2),c(1,3),c(1,4),c(2,3),c(2,4),c(3,4),

          c(1,2,3),c(1,2,4),c(1,3,4),c(2,3,4))

cv.error=rep(0,length(mods))

for(i in 1:length(mods)){

  youtube_mod=youtube[,c(1,1+mods[[i]])]
```

```
  glm.fit=glm(utime~.,data=youtube_mod)

  cv.error[i]=cv.glm(youtube_mod,glm.fit,K=5)$delta[1]

}

print("The cv errors for our models")
```

```
## [1] "The cv errors for our models"
```

```
cv.error
```

```
##  [1] 514.0633 475.8281 260.8026 350.3282 492.8660 251.8281 359.2264 235.2637
##  [9] 345.0736 252.7988 247.0362 324.1670 227.9105 219.6029
```

```
print("the model with lowest cv error")
```

```
## [1] "the model with lowest cv error"
```

```
which.min(cv.error)
```

```
## [1] 14
```

```
glm.fitbest=glm(utime~size+umem+OutputPixels,data=youtube)




shapiro.test(glm.fitbest$residuals)
```

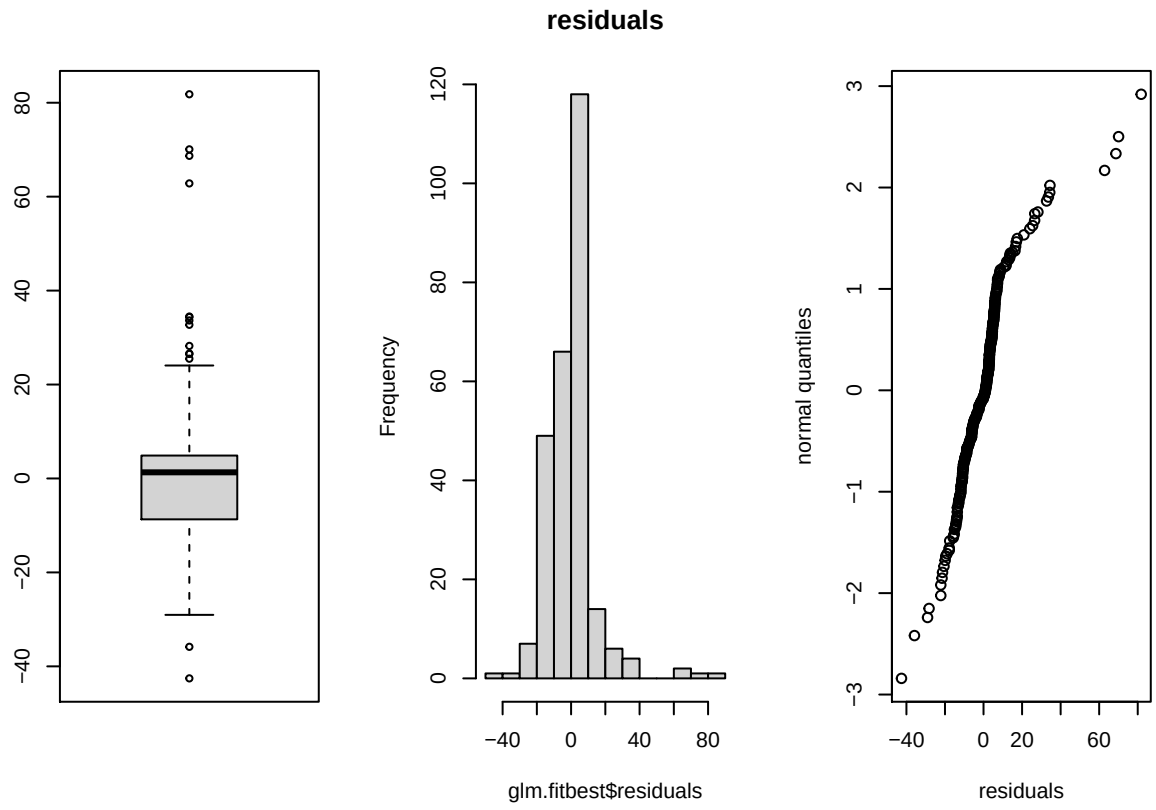
```
##
##  Shapiro-Wilk normality test
##
## data:  glm.fitbest$residuals
## W = 0.838, p-value = 4.047e-16
```

```
par(mfrow=c(1,3))

boxplot(glm.fitbest$residuals)

hist(glm.fitbest$residuals,main="residuals")

qqplot(glm.fitbest$residuals,rnorm(1000),xlab="residuals",ylab="normal quantiles")
```



It seems like our "best" model under 5 fold cross validation is one that predicts upload time from size, computer memory, and output pixels.

Looking at our plot of the residuals, it seems like our residuals have a potentially Leptokurtic distribution, with a slight right skew. This violates our assumption that the residuals would follow a normal distribution without skew.

4.

A

```
vars <- c("npreg","glu", "bp", "skin", "bmi", "ped", "age")
comb.vars <- expand.grid(vars, vars, stringsAsFactors = FALSE)
comb.vars <- comb.vars[comb.vars[,1] != comb.vars[,2],]
i.vars <- apply(comb.vars, 1, paste, collapse = "+")

cv_error<-list(NA)
cost <- function(r, pi = 0) mean(abs(r-pi) > 0.5)
modelfits<-list(NA)
for(i in 1:length(i.vars)) {
  modelformula <- paste("type ~", i.vars[i])
  modelfits[[i]] <- glm(as.formula(modelformula), family = "binomial", data = Pima.tr)
  cv_error[[i]]=cv.glm(Pima.tr,modelfits[[i]], cost = cost ,K=5)$delta[1]
}
cv_error
```

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

```
## [1] 0.285
##
## [[2]]
## [1] 0.325
##
## [[3]]
## [1] 0.29
##
## [[4]]
## [1] 0.29
##
## [[5]]
## [1] 0.27
##
## [[6]]
## [1] 0.295
##
## [[7]]
## [1] 0.27
##
## [[8]]
## [1] 0.255
##
## [[9]]
## [1] 0.235
##
## [[10]]
## [1] 0.24
##
## [[11]]
## [1] 0.245
##
## [[12]]
## [1] 0.255
##
## [[13]]
## [1] 0.3
##
## [[14]]
## [1] 0.265
##
## [[15]]
## [1] 0.365
##
## [[16]]
## [1] 0.315
##
## [[17]]
## [1] 0.33
##
## [[18]]
## [1] 0.295
##
## [[19]]
```

```
## [1] 0.33
##
## [[20]]
## [1] 0.235
##
## [[21]]
## [1] 0.345
##
## [[22]]
## [1] 0.355
##
## [[23]]
## [1] 0.33
##
## [[24]]
## [1] 0.315
##
## [[25]]
## [1] 0.275
##
## [[26]]
## [1] 0.235
##
## [[27]]
## [1] 0.325
##
## [[28]]
## [1] 0.36
##
## [[29]]
## [1] 0.315
##
## [[30]]
## [1] 0.315
##
## [[31]]
## [1] 0.27
##
## [[32]]
## [1] 0.245
##
## [[33]]
## [1] 0.315
##
## [[34]]
## [1] 0.325
##
## [[35]]
## [1] 0.315
##
## [[36]]
## [1] 0.25
##
## [[37]]
```

```
## [1] 0.295
##
## [[38]]
## [1] 0.24
##
## [[39]]
## [1] 0.305
##
## [[40]]
## [1] 0.32
##
## [[41]]
## [1] 0.295
##
## [[42]]
## [1] 0.265
```

```
best_model<-modelfits[[38]]
summary(best_model)
```

```
##
## Call:
## glm(formula = as.formula(modelformula), family = "binomial",
##     data = Pima.tr)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2421  -0.7292  -0.4635   0.7788   2.2913
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.590597   0.950274  -6.935 4.05e-12 ***
## glu          0.032860   0.006375   5.155 2.54e-07 ***
## age          0.052295   0.016796   3.114  0.00185 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 256.41  on 199  degrees of freedom
## Residual deviance: 197.11  on 197  degrees of freedom
## AIC: 203.11
##
## Number of Fisher Scoring iterations: 4
```

The best model which we selected using 5-fold cross validation was one predicting diabetes using glucose levels and age.

My interpretation for the cost function is that it takes the absolute value of the difference between our reported outcome, and our predicted outcome. The understanding that the vector it outputs would be 0 if the guess was the exact same between our predicted and reported outcome, and 1 if the guess was the complete opposite to the predicted outcome. Thus, we are using a threshold value of greater than .5 as a binary outcome of a 'failure', wheras the threshold value lesser than .5 is a binary outcome of a 'successful' prediction.

B

```
glm.probs=predict(best_model, Pima.te ,type="response")
glm.pred=rep("No" ,332)
glm.pred[glm.probs >.5]="Yes"
table(glm.pred,Pima.te$type)
```

```
##
## glm.pred  No Yes
##      No  204  55
##      Yes  19  54
```

```
mean(glm.pred==Pima.te$type)
```

```
## [1] 0.7771084
```

C

```
full_model<-glm(type~., data = Pima.tr, family = "binomial")
step_model<-stepAIC(full_model, direction = "both")
```

```
## Start:  AIC=194.39
## type ~ npreg + glu + bp + skin + bmi + ped + age
##
##         Df Deviance    AIC
## - skin   1   178.40 192.40
## - bp     1   178.46 192.46
## <none>       178.39 194.39
## - npreg  1   180.99 194.99
## - age    1   181.95 195.95
## - bmi    1   182.24 196.24
## - ped    1   186.40 200.40
## - glu    1   205.79 219.79
##
## Step:  AIC=192.4
## type ~ npreg + glu + bp + bmi + ped + age
##
##         Df Deviance    AIC
## - bp     1   178.47 190.47
## <none>       178.40 192.40
## - npreg  1   181.00 193.00
## - age    1   181.97 193.97
## + skin   1   178.39 194.39
## - bmi    1   184.69 196.69
## - ped    1   186.41 198.41
## - glu    1   205.79 217.79
##
## Step:  AIC=190.47
## type ~ npreg + glu + bmi + ped + age
##
##         Df Deviance    AIC
## <none>       178.47 190.47
## - npreg  1   181.08 191.08
## - age    1   182.03 192.03
## + bp     1   178.40 192.40
```

```
## + skin   1   178.46 192.46
## - bmi    1   184.74 194.74
## - ped    1   186.55 196.55
## - glu    1   206.06 216.06
```

```
step_model_1<-stepAIC(glm(type~1, data = Pima.tr, family = "binomial"),scope=formula(full_model),direct
```

```
## Start:  AIC=258.41
## type ~ 1
##
##         Df Deviance    AIC
## + glu    1   207.37 211.37
## + age    1   229.94 233.94
## + bmi    1   239.97 243.97
## + npreg  1   242.03 246.03
## + skin   1   244.70 248.70
## + bp     1   247.55 251.55
## + ped    1   248.11 252.11
## <none>       256.41 258.41
##
## Step:  AIC=211.37
## type ~ glu
##
##         Df Deviance    AIC
## + age    1   197.11 203.11
## + bmi    1   198.47 204.47
## + npreg  1   199.08 205.08
## + ped    1   199.26 205.26
## + skin   1   202.26 208.26
## <none>       207.37 211.37
## + bp     1   205.90 211.90
##
## Step:  AIC=203.11
## type ~ glu + age
##
##         Df Deviance    AIC
## + ped    1   187.10 195.10
## + bmi    1   188.39 196.39
## + skin   1   193.49 201.49
## <none>       197.11 203.11
## + npreg  1   195.55 203.55
## + bp     1   197.10 205.10
##
## Step:  AIC=195.1
## type ~ glu + age + ped
##
##         Df Deviance    AIC
## + bmi    1   181.08 191.08
## + skin   1   184.64 194.64
## + npreg  1   184.74 194.74
## <none>       187.10 195.10
## + bp     1   187.06 197.06
##
## Step:  AIC=191.08
## type ~ glu + age + ped + bmi
```

```
##
##          Df Deviance    AIC
## + npreg  1   178.47 190.47
## <none>       181.08 191.08
## + bp     1   181.00 193.00
## + skin   1   181.06 193.06
##
## Step:  AIC=190.47
## type ~ glu + age + ped + bmi + npreg
##
##        Df Deviance    AIC
## <none>      178.47 190.47
## + bp    1   178.40 192.40
## + skin  1   178.46 192.46
```

```
best_step_model<-glm(type~glu + age +ped +bmi +ped, data = Pima.tr, family = "binomial")

glm.probs=predict(best_step_model, Pima.te ,type="response")
glm.pred=rep("No" ,332)
glm.pred[glm.probs >.5]="Yes"
table(glm.pred,Pima.te$type)
```

```
##
## glm.pred  No Yes
##      No  196  42
##      Yes  27  67
```

```
mean(glm.pred==Pima.te$type)
```

```
## [1] 0.7921687
```

I decided to use a forward and backward stepwise algorithm using AIC as our criterion in order to decided which model is the best logistic regression classification model.

Using this model, we can generate a confusion matrix of our results, and compare it to our previous confusion matrix. Looking at our results, we have an overall lower error rate in our second model found using the stepwise principle. However, while we have a higher specificity we have a lower sensitivity.

D

```
cost <- function(r,p) -2*(sum(log(p)))


modelfits<-list(NA)
for(i in 1:length(i.vars)) {
  modelformula <- paste("type ~", i.vars[i])
  modelfits[[i]] <- glm(as.formula(modelformula), family = "binomial", data = Pima.tr)
  cv_error[[i]]=cv.glm(Pima.tr,modelfits[[i]], cost = cost ,K=5)$delta[1]
}
cv_error
```

```
## [[1]]
## [1] 111.9465
##
## [[2]]
## [1] 93.21729
```

```
##
## [[3]]
## [1] 96.32019
##
## [[4]]
## [1] 99.25117
##
## [[5]]
## [1] 96.96002
##
## [[6]]
## [1] 96.16608
##
## [[7]]
## [1] 112.5454
##
## [[8]]
## [1] 107.2826
##
## [[9]]
## [1] 110.3827
##
## [[10]]
## [1] 112.606
##
## [[11]]
## [1] 111.005
##
## [[12]]
## [1] 110.3765
##
## [[13]]
## [1] 93.76455
##
## [[14]]
## [1] 107.4445
##
## [[15]]
## [1] 92.43428
##
## [[16]]
## [1] 95.79984
##
## [[17]]
## [1] 92.74728
##
## [[18]]
## [1] 96.97121
##
## [[19]]
## [1] 95.24976
##
## [[20]]
## [1] 110.6833
```

```
##
## [[21]]
## [1] 92.57052
##
## [[22]]
## [1] 93.86995
##
## [[23]]
## [1] 92.47752
##
## [[24]]
## [1] 98.67312
##
## [[25]]
## [1] 98.72894
##
## [[26]]
## [1] 111.7658
##
## [[27]]
## [1] 95.17373
##
## [[28]]
## [1] 93.64231
##
## [[29]]
## [1] 94.54691
##
## [[30]]
## [1] 104.0537
##
## [[31]]
## [1] 96.363
##
## [[32]]
## [1] 110.8924
##
## [[33]]
## [1] 93.36916
##
## [[34]]
## [1] 92.66367
##
## [[35]]
## [1] 93.97563
##
## [[36]]
## [1] 100.7291
##
## [[37]]
## [1] 95.98905
##
## [[38]]
## [1] 111.5749
```

```
##
## [[39]]
## [1] 96.73187
##
## [[40]]
## [1] 99.48779
##
## [[41]]
## [1] 104.4211
##
## [[42]]
## [1] 101.1891
```

```
which.min(cv_error)
```

```
## [1] 15
```

```
best_model2<-modelfits[[23]]
summary(best_model2)
```

```
##
## Call:
## glm(formula = as.formula(modelformula), family = "binomial",
##     data = Pima.tr)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7462  -0.8750  -0.6968   1.1949   1.8703
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.57255    0.52214  -4.927 8.35e-07 ***
## ped          1.30250    0.52701   2.471  0.01346 *
## skin         0.04331    0.01455   2.976  0.00292 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 256.41  on 199  degrees of freedom
## Residual deviance: 238.06  on 197  degrees of freedom
## AIC: 244.06
##
## Number of Fisher Scoring iterations: 4
```

I chose to use an extension of the code that was used for 4a, but instead we are using a deviance loss cost function. The results that this alternative loss function found was that the best model was one that predicted diabetes outcome from diabetes pedigree function and skin fold measurements. It is impressive to me that by merely specifying loss function, we were able to come to a different conclusion on what model was superior for our purposes.