Individual Assignment 4

Charlie Ling

9/23/2021

4.7 Exercise, Problem 10 (part f)

10. This question should be answered using the Weekly data set, which is part of the ISLR package. This data is similar in nature to the Smarket data from this chapter's lab, except that it contains 1, 089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

```
rm(list=ls())#release memory
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.0.5
```

```
Weekly=na.omit(Weekly)
```

for context: (d) Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

```
train=(Weekly$Year>=1990)&(Weekly$Year<=2008)
glm.fit=glm(Direction~Lag2,data=Weekly[train,],family = binomial)
glm.probs = predict(glm.fit,Weekly[!train,], type = "response")
glm.pred = rep("Down",length(glm.probs))
glm.pred[glm.probs>0.5]="Up"
table(glm.pred,Weekly[!train,]$Direction)
```

```
##
## glm.pred Down Up
##     Down    9  5
##     Up     34 56
```

```
mean(glm.pred==Weekly[!train,]$Direction)
```

```
## [1] 0.625
```

(f) Repeat (d) using QDA.

```
library(MASS)
qda.fit = qda(Direction~Lag2,data=Weekly[train,])
qda.class = predict(qda.fit,Weekly[!train,])$class
table(qda.class,Weekly[!train,]$Direction)
```

```
##
## qda.class Down Up
##      Down    0  0
##      Up     43 61
```

```
mean(qda.class==Weekly[!train,]$Direction)
```

```
## [1] 0.5865385
```

5.4 Exercise, Problem 8

8. We will now perform cross-validation on a simulated data set.

(a) Generate a simulated data set as follows:

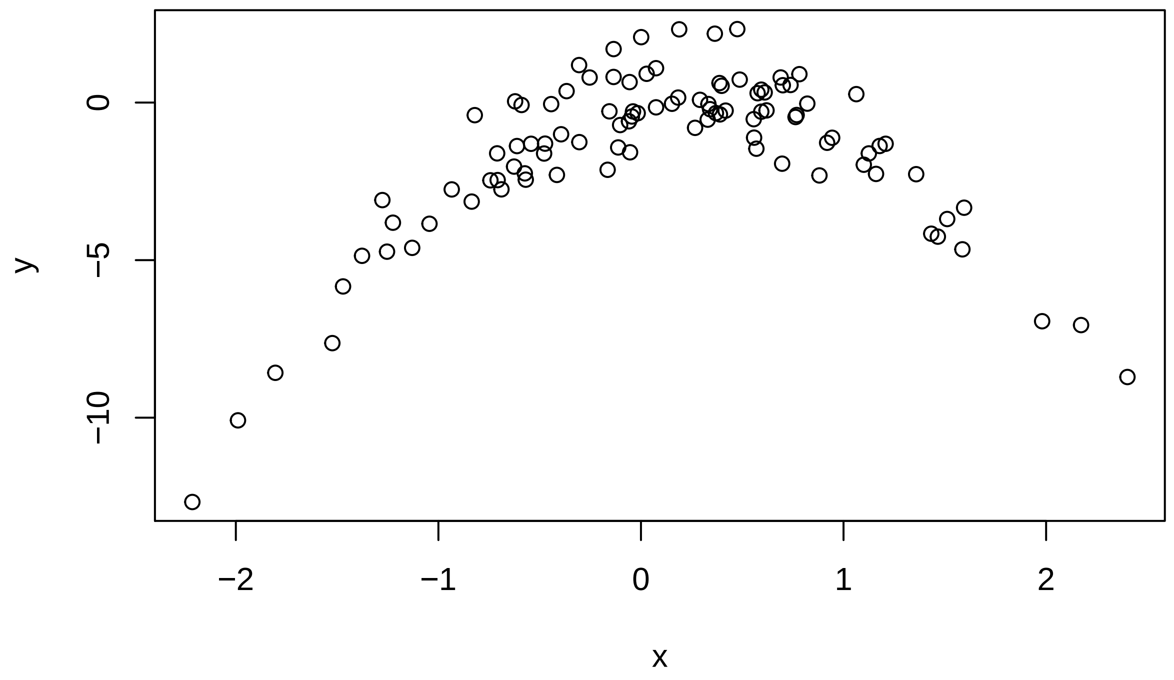
```
set.seed (1)
x=rnorm (100)
y=x-2*x^2+ rnorm (100)
```

In this data set, what is n and what is p? Write out the model used to generate the data in equation form.

```
# n=100, p=1
# y = x - 2*x^2 + e
```

(b) Create a scatterplot of X against Y. Comment on what you find.

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



(c) Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using least squares:

i. Y = B0 + B1X + e
ii. Y = B0 + B1X + B2X^2 + e
iii. Y = B0 + B1X + B2X^2 + B3X^3 + e
iv. Y = B0 + B1X + B2X^2 + B3X^3 + B4X^4 + e.

Note you may find it helpful to use the data.frame() function to create a single data set containing both X and Y.

```r
df=data.frame(y,x)
set.seed (2)
library(boot)
glm.fit1=glm(y~x,data=df)
cv.err1 = cv.glm(df,glm.fit1)
cv.err1$delta
```

```
## [1] 7.288162 7.284744
```

```r
glm.fit2=glm(y~poly(x,2),data=df)
cv.err2 = cv.glm(df,glm.fit2)
cv.err2$delta
```

```
## [1] 0.9374236 0.9371789
```

```r
glm.fit3=glm(y~poly(x,3),data=df)
cv.err3 = cv.glm(df,glm.fit3)
cv.err3$delta
```

```
## [1] 0.9566218 0.9562538
```

```r
glm.fit4=glm(y~poly(x,4),data=df)
cv.err4 = cv.glm(df,glm.fit4)
cv.err4$delta
```

```
## [1] 0.9539049 0.9534453
```

(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 (3)
glm.fit1=glm(y~x,data=df)
cv.err1 = cv.glm(df,glm.fit1)
cv.err1$delta
```

```
## [1] 7.288162 7.284744
```

```r
glm.fit2=glm(y~poly(x,2),data=df)
cv.err2 = cv.glm(df,glm.fit2)
cv.err2$delta
```

```
## [1] 0.9374236 0.9371789
```

```r
glm.fit3=glm(y~poly(x,3),data=df)
cv.err3 = cv.glm(df,glm.fit3)
cv.err3$delta
```

```
## [1] 0.9566218 0.9562538
```

```r
glm.fit4=glm(y~poly(x,4),data=df)
cv.err4 = cv.glm(df,glm.fit4)
cv.err4$delta
```

```
## [1] 0.9539049 0.9534453
```

```r
# The same. LOOCV will always yield the same results because we split 'n' times leaving-out a different
```

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

```
# Model 2 has the smallest LOOCV error. It is what I expect because this model is similar to the model
```

(f) Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in (c) using least squares. Do these results agree with the conclusions drawn based on the cross-validation results?

```
set.seed (1)
summary(glm.fit1)
```

```
##
## Call:
## glm(formula = y ~ x, data = df)
##
## 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)  -1.6254     0.2619  -6.205 1.31e-08 ***
## x             0.6925     0.2909   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
```

```
summary(glm.fit2)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 2), data = df)
##
## 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, 2)1   6.1888     0.9580    6.46 4.18e-09 ***
## poly(x, 2)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
```

```
summary(glm.fit3)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 3), data = df)
##
## 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, 3)1   6.18883    0.96263   6.429 4.97e-09 ***
## poly(x, 3)2 -23.94830    0.96263 -24.878  < 2e-16 ***
## poly(x, 3)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
```

```
summary(glm.fit4)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4), data = df)
##
## 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, 4)1   6.18883    0.95905   6.453 4.59e-09 ***
## poly(x, 4)2 -23.94830    0.95905 -24.971  < 2e-16 ***
## poly(x, 4)3   0.26411    0.95905   0.275    0.784
## poly(x, 4)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
```

```
train=sample(nrow(df),nrow(df)/2,replace = FALSE)
glm.fit1=glm(y~x,data=df)
glm.fit2=glm(y~poly(x,2),data=df)
glm.fit3=glm(y~poly(x,3),data=df)
glm.fit4=glm(y~poly(x,4),data=df)
summary(glm.fit1)
```

```
##
## Call:
## glm(formula = y ~ x, data = df)
##
## 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)  -1.6254     0.2619  -6.205 1.31e-08 ***
## x             0.6925     0.2909   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
```

```
summary(glm.fit2)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 2), data = df)
##
## 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, 2)1   6.1888     0.9580    6.46 4.18e-09 ***
## poly(x, 2)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
```

```
summary(glm.fit3)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 3), data = df)
##
## 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, 3)1   6.18883    0.96263   6.429 4.97e-09 ***
## poly(x, 3)2 -23.94830    0.96263 -24.878  < 2e-16 ***
## poly(x, 3)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
```

```
summary(glm.fit4)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4), data = df)
##
## 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, 4)1   6.18883    0.95905   6.453 4.59e-09 ***
## poly(x, 4)2 -23.94830    0.95905 -24.971  < 2e-16 ***
## poly(x, 4)3   0.26411    0.95905   0.275    0.784
## poly(x, 4)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
```

```
# These results agree with the conclusions drawn based on the cross-validation results
```