Jeffrey Kerley
Jakcqc
March 10th, 2022
HW5

Problem 1:

a)

```
> IBDLine = glm(Default$default ~ Default$balance + Default$income, data = Default,
family = "binomial")
> summary(IBDLine)

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

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 ***
Default$balance  5.647e-03  2.274e-04  24.836  < 2e-16 ***
Default$income   2.081e-05  4.985e-06   4.174 2.99e-05 ***
---
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)

```
> areDef = 0
> areNotDef = 0
> count = 1
> TT = 0
> TF = 0
> FT = 0
> FF = 0
> for(x in IBDprob)
```

```
+ {
+   if(x > .5)
+   {
+     areDef = areDef + 1
+     if(Default$default[count] == "No")
+     {
+       FT = FT + 1
+     }
+     if(Default$default[count] == "Yes")
+     {
+       TT = TT + 1
+     }
+   }
+   if(x<.5)
+   {
+     areNotDef = areNotDef + 1
+     if(Default$default[count] == "No")
+     {
+       TF = TF + 1
+     }
+     if(Default$default[count] == "Yes")
+     {
+       FF = FF + 1
+     }
+   }
+   count = count + 1
+ }
> conMat = matrix(c(TT,FF,FT, TF), nrow = 2)
> conMat

     [P]   [N]
[P]  108   38
[N]  225 9629
> totalRight = conMat[1,1] + conMat[2,2]
> totalWrong = conMat[1,2] + conMat[2,1]
> error = 1 - totalRight/ (totalRight + totalWrong)
> error
[1] 0.0263
```

Error rate of 2.63%.

c)

```
> set.seed(1)
> ##make container class for new sample data
> crossData=sample(1:nrow(Default),size=0.3*nrow(Default))
```

```
> ##generate train data , as -crossData gets 1- ,3, for .7 sample size
> trainData=Default[-crossData,]
> ##default test is .3 so do not need different index
> testData=Default[crossData,]
> head(trainData)
  default student   balance    income
1      No      No  729.5265 44361.625
2      No     Yes  817.1804 12106.135
3      No      No 1073.5492 31767.139
5      No      No  785.6559 38463.496
6      No     Yes  919.5885  7491.559
8      No     Yes  808.6675 17600.451
> head(testData)
     default student   balance   income
1017      No      No  939.0985 45519.02
8004      No     Yes  397.5425 22710.87
4775     Yes      No 1511.6110 53506.94
9725      No      No  301.3194 51539.95
8462      No      No  878.4461 29561.78
4050     Yes      No 1673.4863 49310.33
```

d)

```
> crossIBDLine = glm(trainData$default ~ trainData$balance + trainData$income, data =
trainData, family = binomial)
> summary(crossIBDLine)

Call:
glm(formula = trainData$default ~ trainData$balance + trainData$income,
    family = binomial, data = trainData)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2173  -0.1429  -0.0568  -0.0208   3.7481

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       -1.127e+01  5.250e-01 -21.474   <2e-16 ***
trainData$balance  5.626e-03  2.768e-04  20.329   <2e-16 ***
trainData$income   1.254e-05  6.004e-06   2.088   0.0368 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)
```

Null deviance: 1996.4 on 6999 degrees of freedom
Residual deviance: 1092.7 on 6997 degrees of freedom
AIC: 1098.7

Number of Fisher Scoring iterations: 8

e)

```
> cIBDprob = data.frame(probs = predict(crossIBDLine, type="response", newdata = testData))
> ncIBDprob =  predict(crossIBDLine, type="response", newdata = testData)
##send first 3000 samples of ncIBDprob as it still has the 7000 total samples from training data
##that are useless for testing
> getError(crossIBDLine, testData, ncIBDprob[1:3000])
```

CONFUSION MATRIX

	[P]	[N]
[P]	**1**	**38**
[N]	**106**	**2855**

ERROR RATE

Error rate of 4.8% vs an error rate of 2.63%. This is expected because our data is more diverse and more of a reflection of how generalized this model is. Training and testing on same data will almost always have a lower error rate.

```
[1] 0.048
```

Problem 2:

a)

We see boew that at degree 20 for horsepower we get a coefficient of -.737, at degree 5 we get 13.272, compared to LM, this gives us a coefficient of -0.15788.

```
> res <- numeric(length = 392)
> for (i in 1:392) {
+   lmfit.loocv <- lm(mpg ~ horsepower, data = Auto[-i, ])
+   yhat <- predict(lmfit.loocv, data.frame(horsepower = Auto$horsepower[i]))
+   res[i] <- Auto[i,]$mpg - yhat
+
+   fitter  <- glm(mpg ~poly(horsepower,i), data = Auto)
+   cvLine[i] <- cv.glm(Auto,fitter)$delta[1]
+ }
> mean(res^2)
[1] 1.416441
> fitter$coefficients
         (Intercept)  poly(horsepower, i)1  poly(horsepower, i)2  poly(horsepower, i)3
          23.4459184          -120.1377443            44.0895278            -3.9488485
 poly(horsepower, i)4  poly(horsepower, i)5  poly(horsepower, i)6  poly(horsepower, i)7
```

```
        -5.1878103          13.2721869         -8.5462378          7.9805788
 poly(horsepower, i)8  poly(horsepower, i)9 poly(horsepower, i)10 poly(horsepower, i)11
          2.1727172          -3.9181970         -2.6145722          3.5635748
poly(horsepower, i)12 poly(horsepower, i)13 poly(horsepower, i)14 poly(horsepower, i)15
          1.1450686           0.6040654         -3.8266646         13.4922165
poly(horsepower, i)16 poly(horsepower, i)17 poly(horsepower, i)18 poly(horsepower, i)19
        -14.5099421           9.6577717         -2.2637482         -1.7927856
poly(horsepower, i)20
        -0.7372769
> lmfit.loocv$coefficients
(Intercept)  horsepower
 39.9427431  -0.1578811
```

b)

The data option Auto[-i,], in the lm function is saying to use all data expect for the current row == i, the negative index works as a get all other data, besides the row == -i. For this example in the LOOCV function, it is estimating responses for all 392 overservations. Accomplishing this by removing the current observation from the fit, to judge its response, and then iterating over the rest of the 392 observations.

c)

The data.frame(horsepower = Auto$horsepower[i]) in the predict function is predicting a mpg value, in relation to a specific horsepower value. It is then using the yhat, the predicted value of the current mpg value, and subtracts it from the known mpg value to get the error to predict the MSE value. So, overall this line is using each horsepower value to see what the predicted mpg value is, and then using each value to get a mean of each error amount.

ALL R CODE:

```
## Homework5
library(boot)


Default <- read.csv("Default.csv", stringsAsFactors = TRUE)
Default <- Default[,-1] # Remove the first index column
summary(Default)
## problem 1
#a)
lBDLine = glm(Default$default ~ Default$balance + Default$income, data = Default, family
= binomial)
summary(lBDLine)

#b)
```

```
IBDprob = data.frame(probs = predict(IBDLine, type="response"))
IBDprob =  predict(IBDLine, type="response")

getError(IBDLine, Default, IBDprob)

set.seed(1)

sDefault_default = sample(Default$default, size = 7000)
sDefault_student = sample(Default$student, size = 7000)
sDefault_balance = sample(Default$balance, size = 7000)
sDefault_income = sample(Default$income, size = 7000)

tDefault_default = sample(Default$default[7000:1000], size = 3000)
tDefault_student = sample(Default$student[7000:1000], size = 3000)
tDefault_balance = sample(Default$balance[7000:1000], size = 3000)
tDefault_income = sample(Default$income[7000:1000], size = 3000)


set.seed(1)

##make container class for new sample data
crossData=sample(1:nrow(Default),size=0.3*nrow(Default))
##generate train data , as -crossData gets 1- ,3, for .7 sample size
trainData=Default[-crossData,]
##default test is .3 so do not need different index
testData=Default[crossData,]

head(trainData)
head(testData)

crossIBDLine = glm(trainData$default ~ trainData$balance + trainData$income, data =
trainData, family = binomial)
summary(crossIBDLine)

cIBDprob = data.frame(probs = predict(crossIBDLine, type="response", newdata =
testData))
ncIBDprob =  predict(crossIBDLine, type="response", newdata = testData)

getError(crossIBDLine, testData, ncIBDprob[1:3000])

getError <- function(line, data, probD){
  areDef = 0
  areNotDef = 0
```

```
  count = 1
  TT = 0
  TF = 0
  FT = 0
  FF = 0
  for(x in probD)
  {
   if(x > .5)
   {
    areDef = areDef + 1
    if(data$default[count] == "No")
    {
     FT = FT + 1
    }
    if(data$default[count] == "Yes")
    {
     TT = TT + 1
    }
   }
   if(x<.5)
   {
    areNotDef = areNotDef + 1
    if(data$default[count] == "No")
    {
     TF = TF + 1
    }
    if(data$default[count] == "Yes")
    {
     FF = FF + 1
    }
   }
   count = count + 1
  }
  conMat = matrix(c(TT,FF,FT, TF), nrow = 2)
  print(conMat)

  totalRight = conMat[1,1] + conMat[2,2]
  totalWrong = conMat[1,2] + conMat[2,1]
  error = 1 - totalRight/ (totalRight + totalWrong)
  print(error)
}
```

```r
#problem 2
Auto <- read.csv("Auto.csv", na.strings ="?") # With the option, R recognizes ? as NA.
Auto <- na.omit(Auto) # Remove data rows including NA.
Auto$origin <- as.factor(Auto$origin) # Coerce the type of origin into factor


#a)
res <- numeric(length = 392)
for (i in 1:392) {
  lmfit.loocv <- lm(mpg ~ horsepower, data = Auto[-i, ])
  yhat <- predict(lmfit.loocv, data.frame(horsepower = Auto$horsepower[i]))
  res[i] <- Auto[i,]$mpg - yhat
}
mean(res^2)

cvLine <- rep(0, 20)
for(i in 1:20)
{
  fitter  <- glm(mpg ~poly(horsepower,i), data = Auto)
  cvLine[i] <- cv.glm(Auto,fitter)$delta[1]
}
lmfit.loocv$coefficients
fitter$coefficients
cvLine
```