Homework 5

Drew Dahlquist

3/8/2022

1)

```
Default = read.csv("Default.csv", stringsAsFactors = TRUE)
Default = Default[,-1] # Remove the first index column
summary(Default)
```

```
##  default    student       balance           income
##  No :9667   No :7056   Min.   :   0.0   Min.   :  772
##  Yes: 333   Yes:2944   1st Qu.: 481.7   1st Qu.:21340
##                        Median : 823.6   Median :34553
##                        Mean   : 835.4   Mean   :33517
##                        3rd Qu.:1166.3   3rd Qu.:43808
##                        Max.   :2654.3   Max.   :73554
```

(a)

```
glm.fit = glm(default ~ income + balance, family=binomial, data=Default)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = default ~ income + balance, 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 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 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: 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)

```
glm.probs = predict(glm.fit,type="response")
glm.pred=rep("No", length(Default$default))
glm.pred[glm.probs > 0.5] = "Yes"
table(glm.pred, Default$default)
```

```
##
## glm.pred   No  Yes
##      No  9629  225
##      Yes   38  108
```

Error rate is 37.0228137.

(c)

```
set.seed(1)
size=length(Default$default)
train=sample(size, 0.7*size)
```

(d)

```
glm.fit = glm(default ~ income + balance, family=binomial, data=Default, subset=train)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = binomial,
##     data = Default, subset = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4481  -0.1402  -0.0561  -0.0211   3.3484
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.167e+01  5.214e-01 -22.379  < 2e-16 ***
## income       2.560e-05  6.012e-06   4.258 2.06e-05 ***
## balance      5.574e-03  2.678e-04  20.816  < 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: 2030.3  on 6999  degrees of freedom
## Residual deviance: 1079.6  on 6997  degrees of freedom
## AIC: 1085.6
##
## Number of Fisher Scoring iterations: 8
```

(e)

```
glm.probs = predict(glm.fit,type="response")[-train]
glm.pred=rep("No", length(Default$default[-train]))
glm.pred[glm.probs > 0.5] = "Yes"
table(glm.pred, Default$default[-train])
```

```
##
```

```
## glm.pred   No  Yes
##       No  2846  100
##       Yes   52    2
```

Error rate is 18.7368421.

The test error rate obtained from the model trained on a train/test split is one half that of the training error rate obtained from the model trained only on training data in part (b). This is likely due to overfitting of the model, since training error rate can be made arbitrarily small, whereas the test error is assumed to not be able to go below some non-zero threshold due to noise.

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)

```
# LOOCV
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)
```

```
## [1] 24.23151
```

The outcome is exactly the same as the standard CV estimate obtained from `cv.glm()`.

(b)

The option `data = Auto[-i, ]` is exactly what is performing "leave one out" part of LOOCV for us. It is selecting all indices from the `Auto` data except the i'th value (i.e. it's leaving the i'th value out).

(c)

The input `data.frame(horsepower = Auto$horsepower[i])` is telling `predict()` to make a prediction on the i'th value of `Auto$horsepower`. This combined with what is explained in part (b), completes the LOOCV by fitting a model on all but one data point, then using the leftover data point to assess the error.