# STATS 415 - Homework 5

*Marian L. Schmidt*

*February 24, 2016*

**1. The textbook describes that the `cv.glm()`function can be used in order to compute the LOOCV test error estimate. Alternatively, one could compute those quantities using just the `glm()` and `predict.glm()` functions, and a for loop. You will now take this approach in order to compute the LOOCV error for a simple logistic regression model on the Weekly data set.**

```
# Load ISLR Library
library(ISLR)

# Set the Seed for reproducibility
set.seed(232)

# Look at what the data structure looks like
# How many rows and columns?
nrow(Weekly) # Number of Rows
```

```
## [1] 1089
```

```
ncol(Weekly) # Number of columns
```

```
## [1] 9
```

```
#What does the data look like?
head(Weekly)
```

```
##   Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today Direction
## 1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
## 2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
## 3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514        Up
## 4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712        Up
## 5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178        Up
## 6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372      Down
```

```
# What type of data do we have?
str(Weekly)
```

```
## 'data.frame':    1089 obs. of  9 variables:
##  $ Year     : num  1990 1990 1990 1990 1990 1990 1990 1990 1990 1990 ...
##  $ Lag1     : num  0.816 -0.27 -2.576 3.514 0.712 ...
##  $ Lag2     : num  1.572 0.816 -0.27 -2.576 3.514 ...
##  $ Lag3     : num  -3.936 1.572 0.816 -0.27 -2.576 ...
##  $ Lag4     : num  -0.229 -3.936 1.572 0.816 -0.27 ...
##  $ Lag5     : num  -3.484 -0.229 -3.936 1.572 0.816 ...
##  $ Volume   : num  0.155 0.149 0.16 0.162 0.154 ...
##  $ Today    : num  -0.27 -2.576 3.514 0.712 1.178 ...
##  $ Direction: Factor w/ 2 levels "Down","Up": 1 1 2 2 2 1 2 2 2 1 ...
```

### (a) Fit a logistic regression model that predicts Direction using Lag1 and Lag2.

```
# Fit the logistic regression with all the data
logistic_all <-  glm(Direction ~ Lag1 + Lag2, data = Weekly, family=binomial)
summary(logistic_all)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = Weekly)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.623  -1.261   1.001   1.083   1.506
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22122    0.06147   3.599 0.000319 ***
## Lag1        -0.03872    0.02622  -1.477 0.139672
## Lag2         0.06025    0.02655   2.270 0.023232 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1488.2  on 1086  degrees of freedom
## AIC: 1494.2
##
## Number of Fisher Scoring iterations: 4
```

### (b) Fit a logistic regression model that predicts Direction using Lag1 and Lag2 using all but the first observation.

```
# Fit the logistic regression with all the data except one observation
logistic_one <-  glm(Direction ~ Lag1 + Lag2, data = Weekly[-1,], family = binomial)
summary(logistic_one)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = Weekly[-1,
##     ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6258  -1.2617   0.9999   1.0819   1.5071
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22324    0.06150   3.630 0.000283 ***
## Lag1        -0.03843    0.02622  -1.466 0.142683
## Lag2         0.06085    0.02656   2.291 0.021971 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1494.6  on 1087  degrees of freedom
## Residual deviance: 1486.5  on 1085  degrees of freedom
## AIC: 1492.5
##
## Number of Fisher Scoring iterations: 4
```

**(c) Use the model from (b) to predict the direction of the first observation. You can do this by predicting that the first observation will go up if `Pr(Direction="Up" | Lag1, Lag2) > 0.5`. Was this observation correctly classified?**

```
# Create a prediction regarding the market
logistic_prediction_up <- predict.glm(logistic_one, Weekly[1,], type = "response") > 0.05

# Is the market predicted to go up for the first observation?
logistic_prediction_up
```

```
##    1
## TRUE
```

```
# Does the market actually go up?
true_up <- Weekly[1, ]$Direction == "Up"

# Does the prediction match the market reality?
logistic_prediction_up != true_up
```

```
##    1
## TRUE
```

*This observation was incorrectly classified. The prediction for the first observation is "Up", however, the actual observation was "Down".*

**(d) Write a for loop from `i=1` to `i=n`, where `n` is the number of observations in the data set, that performs each of the following steps:**

1. **Fit a logistic regression model using all but the ith observation to predict Direction using Lag1 and Lag2.**
2. **Compute the posterior probability of the market moving up for the ith observation.**
3. **Use the posterior probability for the ith observation in order to predict whether or not the market moves up.**
4. **Determine whether or not an error was made in predicting the direction for the ith observation. If an error was made, then indicate this as a 1, and otherwise indicate it as a 0.**

```
# Number of iterations in for loop
n <- nrow(Weekly); n
```

```
## [1] 1089
```

```
# Create 0s for error that we will fill in with ones
prediction_error <- rep(0, n)
for (i in 1:n){
  # Step 1: Run a logistic regression leaving one observation point out
  logistic_regression <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-i,], family = binomial)
  # Step 2: Create a prediction on the one observation not included in the logistic regression
  market_pred_up <- predict.glm(logistic_regression, Weekly[i,], type = "response") > 0.05
  # Step 3: Pull out all
  market_true_up <- Weekly[i, ]$Direction == "Up"
  # Step 4: If an error was made in our prediction, add a "1" to error vector
  if(market_pred_up != market_true_up) prediction_error[i] <- 1
}
prediction_error
```

```
##    [1] 1 1 0 0 0 1 0 0 0 1 1 0 0 0 1 0 1 0 1 0 0 0 1 1 1 1 1 1 0 0 1 1 1 0
##   [35] 1 0 1 0 0 0 1 0 0 1 0 1 1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 1 0 0
##   [69] 1 0 1 1 0 0 0 1 0 1 1 0 0 1 1 0 1 1 0 0 1 0 0 1 1 1 0 0 0 0 0 1 0 1
##  [103] 1 0 0 1 0 1 0 0 1 1 0 0 1 0 0 1 0 0 1 1 1 1 0 0 0 1 0 1 0 1 1 0 0 0
##  [137] 1 1 1 0 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 1 0 0 0 0 1 1 0 0 1 0 0 1
##  [171] 0 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 1 0 0 1 0 0 1 0
##  [205] 0 1 0 1 0 1 1 1 0 0 1 1 0 1 0 0 1 1 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 1
##  [239] 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0
##  [273] 0 0 1 0 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 1 1
##  [307] 0 0 1 0 0 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0 0 0 1 0 1 0 0 1 1 1 1 0 1 0
##  [341] 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 1 1 1
##  [375] 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 0 1 0 0 1 1 1 0 0 0 1 0
##  [409] 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 1
##  [443] 1 1 0 1 1 0 0 0 1 1 0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 0 1 1 0 1 0 0 0 1
##  [477] 0 0 1 0 1 0 1 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0
##  [511] 1 0 1 0 0 0 1 0 1 1 0 1 1 1 0 1 0 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 0 0
##  [545] 1 1 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 0 1 1 0 1 1 0 1 0 0 0 1 1 1 1 1 1
##  [579] 1 1 0 1 0 0 0 0 1 0 1 1 0 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 0 0 1 0 1 0
##  [613] 0 0 1 0 1 0 0 0 1 1 0 1 1 0 1 0 0 0 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1
##  [647] 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 0 0 0 1 1 0 1 0 0 1 1 1 1 0 0 1
##  [681] 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 0 1
##  [715] 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0 1 1 1 1 0 0 0 1
##  [749] 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 0 0 1 0 0 0 1 1 1 0 0 0
##  [783] 1 0 0 1 1 1 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 1 1 0 1
##  [817] 1 1 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 0 0 0 1 1 0 1 1 0 1 0 1 0 1 1 0 0
##  [851] 1 1 1 0 1 1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 1 1 0 0 1 0 1 0 1 1
##  [885] 0 1 0 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1 1 1 0 1 0 1 1 0 0
##  [919] 0 0 0 1 0 1 1 0 1 0 0 1 0 1 1 1 1 0 0 1 0 0 1 1 1 0 1 0 1 0 0 0 1 0
##  [953] 1 0 1 1 1 1 1 1 0 1 0 0 0 1 1 1 0 0 1 1 1 0 1 0 1 1 1 0 1 0 0 1 0 1
##  [987] 1 1 1 0 1 1 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 1 1 1 0 0 0 0 1 0 0 1
## [1021] 0 0 1 1 0 0 1 1 0 0 1 0 0 0 1 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 0 1 0 1
## [1055] 1 0 1 0 1 0 0 1 1 0 1 0 1 0 1 1 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0
## [1089] 0
```

```
sum(prediction_error)
```

```
## [1] 484
```

*In the above example of LOOCV, there are 484 misclassified predictions out of 1089.*

**(e) Take the average of the `n` numbers obtained in (d)iv in order to obtain the LOOCV estimate for the test error. Comment on the results.**

```
mean(prediction_error)
```

```
## [1] 0.4444444
```

```
mean(prediction_error) * 100
```

```
## [1] 44.44444
```

*The LOOCV estimate for the test error rate is 44.4444444%*