STATS 415 Lab 5

Nick Seewald

9 February 2018

Partially adapted from http://www-bcf.usc.edu/~gareth/ISL/Chapter%204%20Lab.txt

1 Today's Objectives

1. Learn how to perform logistic regression in R
2. Learn how to interpret parameters in logistic regression models
3. Build a classifier using logistic regression and compare it to classifiers constructed using other methods

2 Building a classifier with logistic regression

We'll use the `Smarket` data from the `ISLR` package. This is daily data on percentage returns for the S&P 500 stock index for 1200 days from 2001 to 2005.

Our goal: Use recent historical data to predict whether the stock market will go up or down (i.e., try to get rich).

2.1 Exploratory data analysis

We start by loading the dataset and looking at what information it contains.

```
library(ISLR)
data(Smarket)
str(Smarket)
```

```
## 'data.frame':    1250 obs. of  9 variables:
##  $ Year     : num  2001 2001 2001 2001 2001 ...
##  $ Lag1     : num  0.381 0.959 1.032 -0.623 0.614 ...
##  $ Lag2     : num  -0.192 0.381 0.959 1.032 -0.623 ...
##  $ Lag3     : num  -2.624 -0.192 0.381 0.959 1.032 ...
##  $ Lag4     : num  -1.055 -2.624 -0.192 0.381 0.959 ...
##  $ Lag5     : num  5.01 -1.055 -2.624 -0.192 0.381 ...
##  $ Volume   : num  1.19 1.3 1.41 1.28 1.21 ...
##  $ Today    : num  0.959 1.032 -0.623 0.614 0.213 ...
##  $ Direction: Factor w/ 2 levels "Down","Up": 2 2 1 2 2 2 1 2 2 2 ...
```

Let's look at the variables in more detail.

```
summary(Smarket)
```

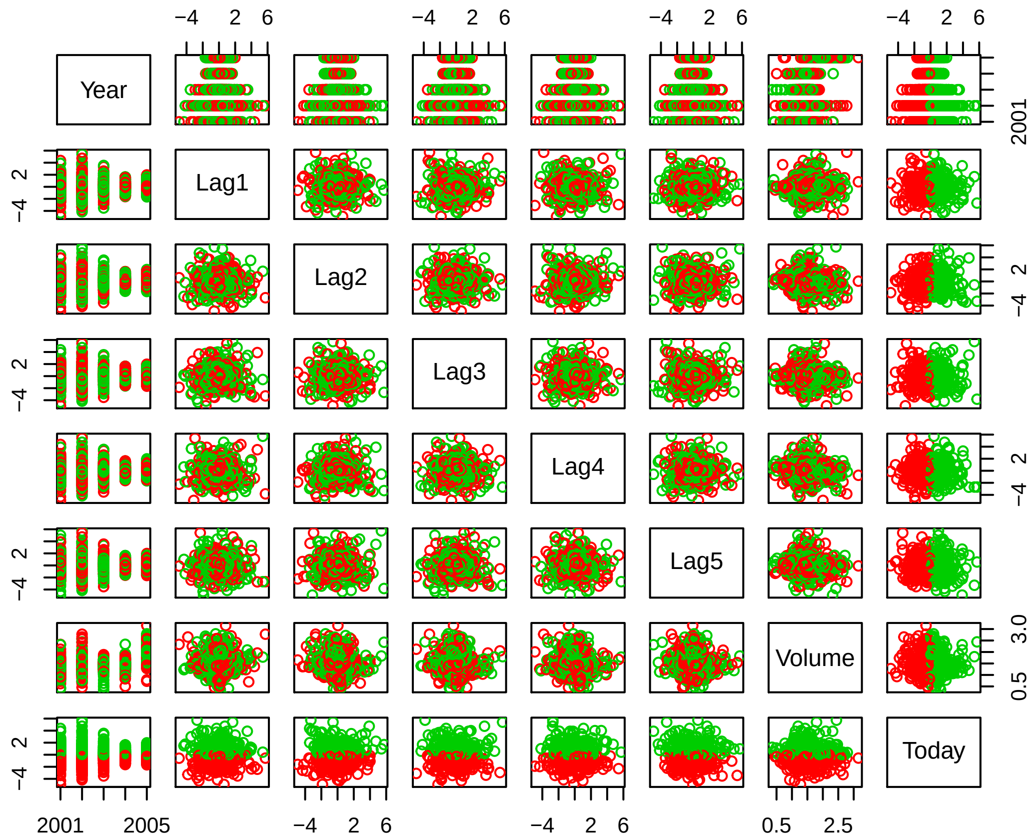
```
##       Year           Lag1                Lag2          
##  Min.   :2001   Min.   :-4.922000   Min.   :-4.922000  
##  1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500  
##  Median :2003   Median : 0.039000   Median : 0.039000  
##  Mean   :2003   Mean   : 0.003834   Mean   : 0.003919  
##  3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750  
```

```
##  Max.   :2005   Max.   : 5.733000   Max.   : 5.733000
##       Lag3                Lag4                Lag5         
##  Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.92200  
##  1st Qu.:-0.640000   1st Qu.:-0.640000   1st Qu.:-0.64000  
##  Median : 0.038500   Median : 0.038500   Median : 0.03850  
##  Mean   : 0.001716   Mean   : 0.001636   Mean   : 0.00561  
##  3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.59700  
##  Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.73300  
##      Volume           Today           Direction 
##  Min.   :0.3561   Min.   :-4.922000   Down:602  
##  1st Qu.:1.2574   1st Qu.:-0.639500   Up  :648  
##  Median :1.4229   Median : 0.038500             
##  Mean   :1.4783   Mean   : 0.003138             
##  3rd Qu.:1.6417   3rd Qu.: 0.596750             
##  Max.   :3.1525   Max.   : 5.733000
```

Question: Look at the lag variables. Does anything surprise you? What's the explanation for this?

Let's make some plots. Notice we exclude `Direction` because it's categorical.

```r
pairs(Smarket[, -9], col = c("red", "green3")[Smarket$Direction])
```



It doesn't look like there's much correlation among any of the `Lags`, or between any of the `Lags` and `Today`. This makes sense – if it were that easy to predict the stock market, we'd all be rich! Let's confirm this numerically by looking at sample correlations:

```
round(cor(Smarket[, -9]), 3)
```

```
##          Year   Lag1   Lag2   Lag3   Lag4   Lag5 Volume  Today
## Year    1.000  0.030  0.031  0.033  0.036  0.030  0.539  0.030
## Lag1    0.030  1.000 -0.026 -0.011 -0.003 -0.006  0.041 -0.026
## Lag2    0.031 -0.026  1.000 -0.026 -0.011 -0.004 -0.043 -0.010
## Lag3    0.033 -0.011 -0.026  1.000 -0.024 -0.019 -0.042 -0.002
## Lag4    0.036 -0.003 -0.011 -0.024  1.000 -0.027 -0.048 -0.007
## Lag5    0.030 -0.006 -0.004 -0.019 -0.027  1.000 -0.022 -0.035
## Volume  0.539  0.041 -0.043 -0.042 -0.048 -0.022  1.000  0.015
## Today   0.030 -0.026 -0.010 -0.002 -0.007 -0.035  0.015  1.000
```

The `cor()` function returns a matrix containing the pairwise sample correlations for every variable in the dataset given to it as an argument. It can only handle quantitative variables, so we remove `Direction`.

2.2 A quick theory review

Remember that we use logistic regression to fit a model to binary outcomes. These can often be thought of as yes/no answers to questions like "Did the Red Wings win the game?", "Did you smoke one or more cigarettes in the last 24 hours?", or "Did the stock market go up or down today?".

The logistic regression model has the form

$$\log\left(\frac{P(Y=1 \mid X=x)}{1-P(Y=1 \mid X=x)}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p.$$

The left-hand side of the model is called the *logit* (or *log odds*), which is linear in X.

2.3 Fitting a logistic regression model

Our goal is to fit a logistic regression model to predict `Direction` using the `Lag` variables and `Volume`.

Question: Why don't we want to include `Today` in our model?

To assess how well our model works, we'll train on data from 2001-2004 and test our classifer on data from 2005.

```
trainData <- subset(Smarket, Year <= 2004)
testData  <- subset(Smarket, Year == 2005)
```

To fit a logistic regression, we use the `glm()` function. GLM stands for *generalized linear model*, which is a class of models that includes logistic regression (and many others that are outside the scope of this course).

```
mod1 <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
            data = trainData, family = binomial)
```

Let's review this syntax:

- It's very similar to `lm()`! The key difference is the addition of the `family` argument. Because there are many different types of GLM, we need to tell R to run a logistic regression. Setting `family = binomial` does this.

Just like with `lm` objects, we can use `summary()` to get information about the fit of the model:

```
summary(mod1)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = trainData)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.302  -1.190   1.079   1.160   1.350
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.191213   0.333690   0.573    0.567
## Lag1        -0.054178   0.051785  -1.046    0.295
## Lag2        -0.045805   0.051797  -0.884    0.377
## Lag3         0.007200   0.051644   0.139    0.889
## Lag4         0.006441   0.051706   0.125    0.901
## Lag5        -0.004223   0.051138  -0.083    0.934
## Volume      -0.116257   0.239618  -0.485    0.628
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1383.3  on 997  degrees of freedom
## Residual deviance: 1381.1  on 991  degrees of freedom
## AIC: 1395.1
##
## Number of Fisher Scoring iterations: 3
```

Remember that `Direction` is a factor variable with levels DownandUp. How do we know how R coded the dummy variable for `Direction`?

```r
contrasts(trainData$Direction)
```

```
##      Up
## Down  0
## Up    1
```

This tells us that R chose `Down` to be 0 and `Up` to be 1. Therefore, the fitted values in our model correspond to $P(\texttt{Direction} = \texttt{Up} \mid X = x)$, where X is the vector of predictors.

2.4 Parameter interpretation

What do the parameters in a logistic regression model *mean*? Let's look at a simple model with 2 predictors ($p = 2$):

$$\log\left(\frac{P(Y = 1 \mid X = x)}{1 - P(Y = 1 \mid X = x)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2.$$

We'll start with β_0. Remember that the game here is to get each parameter by itself on one side of the equation, without any other parameters. We can do this for β_0 by setting $x_1 = x_2 = 0$. Then we have

$$\begin{aligned}\beta_0 &= \log\left(\frac{P(Y = 1 \mid X_1 = 0, X_2 = 0)}{1 - P(Y = 1 \mid X_1 = 0, X_2 = 0)}\right) \\ e^{\beta_0} &= \text{odds}(Y = 1 \mid X_1 = 0, X_2 = 0)\end{aligned}$$

Similarly, we can do this for β_1, β_2.

- **Step 1:** Write out model equations for $X_1 = x_1$ and $X_1 = x_1 + 1$.

- **Step 2:** Subtract to isolate β_1.

- **Step 3:** Exponentiate.

Question: Is this an additive effect?

Question: How do we interpret β_1 in words?

2.5 Prediction using logistic regression

As before, we can use the `predict()` function to get fitted values for a logistic regression model.

```
pred <- predict(mod1, testData)
```

Note that the output of `predict()` is *NOT* the predicted probability! Recall the form of the logistic regression model:

$$\log\left(\frac{P(Y=1 \mid X=x)}{1-P(Y=1 \mid X=x)}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \epsilon.$$

Question: What are the units of the output of `predict()`?

To classify observations, we have to transform the output to be on the probability scale. The inverse of the *logit* function is called the *expit* function:

$$\text{logit}(x) := \log\left(\frac{x}{1-x}\right) \qquad\qquad \text{logit}^{\leftarrow}(x) = \text{expit}(x) := \frac{e^x}{1+e^x}$$

There are two ways to take the expit of something in R. The first is to code it yourself:

```
expit <- function(x) exp(x) / (1 + exp(x))
expit(.5)
```

```
## [1] 0.6224593
```

and the second is to use `binomial()$linkinv()`:

```r
binomial()$linkinv(.5)
```

```
## [1] 0.6224593
```

```r
expit <- function(x) binomial()$linkinv(x)
expit(.5)
```

```
## [1] 0.6224593
```

We can get predicted probabilities by applying `expit` to our `pred` variable. If the predicted probability is greater than 0.5, we classify the observation as `Up`; otherwise, we classify it as `Down`.

```r
# Get predicted probabilities
predProbs <- expit(pred)
# Create a vector of all "Down"s (we'll replace the entries that should be
# classified as "Up" in the next step)
testPrediction <- rep("Down", nrow(testData))
# Replace "Down" with "Up" if the predicted probability is greater than .5
testPrediction[predProbs > .5] <- "Up"
# Create a confusion matrix
table(testPrediction, testData$Direction, dnn = c("Predicted", "Actual"))
```

```
##          Actual
## Predicted Down Up
##      Down   77 97
##      Up     34 44
```

We compute the test error:

```r
round(mean(testPrediction != testData$Direction), 2)
```

```
## [1] 0.52
```

Alternatively, using the confusion matrix:

```r
round((97 + 34) / (77 + 97 + 34 + 44), 2)
```

```
## [1] 0.52
```

Question: How does our logistic regression classifier perform on this data? Does that make sense?

Let's try to improve this. None of the predictors in our original logistic regression model are significantly different from zero, but the p values for `Lag3`, `Lag4`, `Lag5`, and `Volume` are exceptionally high. Let's take them out and see if we can improve.

```r
# Fit new model
mod2 <- glm(Direction ~ Lag1 + Lag2, data = trainData, family = binomial)
summary(mod2)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = trainData)
##
```

```
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.345  -1.188   1.074   1.164   1.326
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.03222    0.06338   0.508    0.611
## Lag1        -0.05562    0.05171  -1.076    0.282
## Lag2        -0.04449    0.05166  -0.861    0.389
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1383.3  on 997  degrees of freedom
## Residual deviance: 1381.4  on 995  degrees of freedom
## AIC: 1387.4
##
## Number of Fisher Scoring iterations: 3
```

```r
# Get predicted probabilities
predProbs2 <- expit(predict(mod2, testData))
testPrediction2 <- rep("Down", nrow(testData))
testPrediction2[predProbs2 > .5] <- "Up"
# Confusion matrix
table(testPrediction2, testData$Direction, dnn = c("Predicted", "Actual"))
```

```
##          Actual
## Predicted Down  Up
##      Down   35  35
##      Up     76 106
```

```r
round(mean(testPrediction2 != testData$Direction), 2)
```

```
## [1] 0.44
```

Question: What does the confusion matrix tell us about our predictive abilities?

3 Logistic regression vs. Other Classification Methods

Let's compare our (simplified) logistic regression classifier to classifiers produced by LDA, QDa, and KNN.

3.1 LDA

We'll start with LDA.

```r
library(MASS)
lda.fit <- lda(Direction ~ Lag1 + Lag2, data = trainData)
lda.fit
```

```
## Call:
## lda(Direction ~ Lag1 + Lag2, data = trainData)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
##
## Coefficients of linear discriminants:
##             LD1
## Lag1 -0.6420190
## Lag2 -0.5135293
```

```
lda.pred <- predict(lda.fit, testData)
lda.class <- lda.pred$class
table(lda.class, testData$Direction, dnn = c("Predicted", "Actual"))
```

```
##          Actual
## Predicted Down  Up
##      Down   35  35
##      Up     76 106
```

```
round(mean(lda.class != testData$Direction), 2)
```

```
## [1] 0.44
```

Question: How does this test error compare to that of logistic regression?

3.2 QDA

```
qda.fit <- qda(Direction ~ Lag1 + Lag2, data = trainData)
qda.fit
```

```
## Call:
## qda(Direction ~ Lag1 + Lag2, data = trainData)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
```

```r
qda.class <- predict(qda.fit, testData)$class
table(qda.class, testData$Direction, dnn = c("Predicted", "Actual"))
```

```
##           Actual
## Predicted Down  Up
##      Down   30  20
##      Up     81 121
```

```r
mean(qda.class != testData$Direction)
```

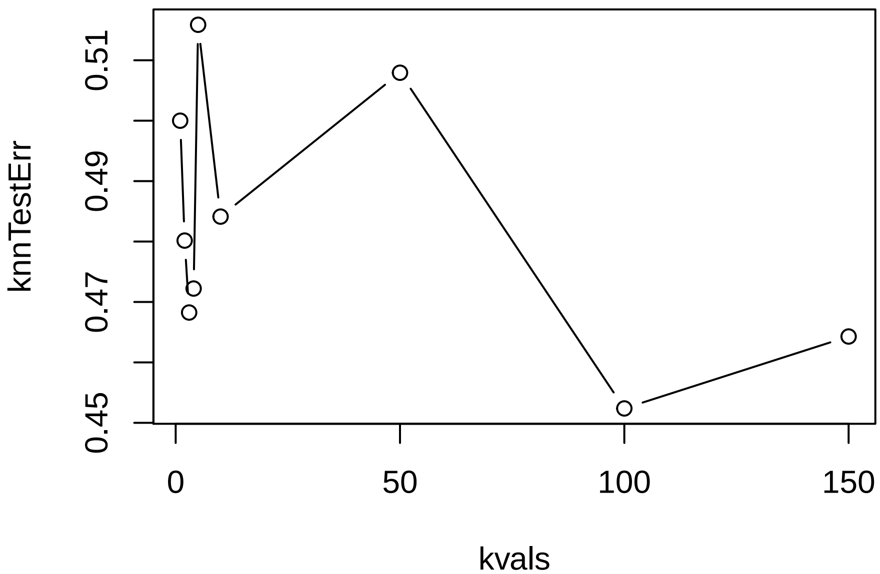
```
## [1] 0.4007937
```

Question: How does this test error compare to those of logistic regression and LDA?

3.3 KNN

```r
library(class)
trainX <- as.matrix(trainData[c("Lag1", "Lag2")])
testX  <- as.matrix(testData[c("Lag1", "Lag2")])

set.seed(1)
kvals <- c(1:5, 10, 50, 100, 150)
knnTestErr <- vector(length = length(kvals))
for (i in 1:length(kvals)) {
  knn.pred <- knn(train = trainX, test = testX, cl = trainData$Direction, k=kvals[i])
  knnTestErr[i] <- mean(knn.pred != testData$Direction)
}
plot(knnTestErr ~ kvals, type = "b")
```



It seems like $k=$100 gives us the lowest test error. Let's examine the prediction further.

```
kmin <- kvals[which.min(knnTestErr)]
knn.pred <- knn(train = trainX, test = testX, cl = trainData$Direction, k = kmin)
table(knn.pred, testData$Direction, dnn = c("Predicted", "Actual"))
```

```
##           Actual
## Predicted Down Up
##      Down   47 55
##      Up     64 86
```

```
mean(knn.pred != testData$Direction)
```

```
## [1] 0.4722222
```

Question: How does this fit compare to the previous classifiers? Which is the best for this data?