

4.7 Lab: Classification Methods

4.7.1 The Stock Market Data

We will begin by examining some numerical and graphical summaries of the `Smarket` data, which is part of the `ISLR2` library. This data set consists of percentage returns for the S&P 500 stock index over 1,250 days, from the beginning of 2001 until the end of 2005. For each date, we have recorded the percentage returns for each of the five previous trading days, `Lag1` through `Lag5`. We have also recorded `Volume` (the number of shares traded on the previous day, in billions), `Today` (the percentage return on the date in question) and `Direction` (whether the market was `Up` or `Down` on this date). Our goal is to predict `Direction` (a qualitative response) using the other features.

```
> library(ISLR2)
> names(Smarket)
[1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"
[6] "Lag5"      "Volume"    "Today"     "Direction"
> dim(Smarket)
[1] 1250      9
> summary(Smarket)
```

Year		Lag1	Lag2	
Min. :	2001	Min. : -4.92200	Min. :	-4.92200
1st Qu. :	2002	1st Qu. : -0.63950	1st Qu. :	-0.63950
Median :	2003	Median : 0.03900	Median :	0.03900
Mean :	2003	Mean : 0.00383	Mean :	0.00392
3rd Qu. :	2004	3rd Qu. : 0.59675	3rd Qu. :	0.59675
Max. :	2005	Max. : 5.73300	Max. :	5.73300

Lag3		Lag4	Lag5	
Min. :	-4.92200	Min. : -4.92200	Min. :	-4.92200
1st Qu. :	-0.64000	1st Qu. : -0.64000	1st Qu. :	-0.64000
Median :	0.03850	Median : 0.03850	Median :	0.03850
Mean :	0.00172	Mean : 0.00164	Mean :	0.00561
3rd Qu. :	0.59675	3rd Qu. : 0.59675	3rd Qu. :	0.59700
Max. :	5.73300	Max. : 5.73300	Max. :	5.73300

Volume		Today	Direction
Min. :	0.356	Min. : -4.92200	Down:602
1st Qu. :	1.257	1st Qu. : -0.63950	Up :648
Median :	1.423	Median : 0.03850	
Mean :	1.478	Mean : 0.00314	
3rd Qu. :	1.642	3rd Qu. : 0.59675	
Max. :	3.152	Max. : 5.73300	

```
> pairs(Smarket)
```

The `cor()` function produces a matrix that contains all of the pairwise correlations among the predictors in a data set. The first command below gives an error message because the `Direction` variable is qualitative.

```
> cor(Smarket)
Error in cor(Smarket) : 'x' must be numeric
> cor(Smarket[, -9])
```

	Year	Lag1	Lag2	Lag3	Lag4	Lag5
Year	1.0000	0.02970	0.03060	0.03319	0.03569	0.02979
Lag1	0.0297	1.00000	-0.02629	-0.01080	-0.00299	-0.00567
Lag2	0.0306	-0.02629	1.00000	-0.02590	-0.01085	-0.00356
Lag3	0.0332	-0.01080	-0.02590	1.00000	-0.02405	-0.01881
Lag4	0.0357	-0.00299	-0.01085	-0.02405	1.00000	-0.02708
Lag5	0.0298	-0.00567	-0.00356	-0.01881	-0.02708	1.00000
Volume	0.5390	0.04091	-0.04338	-0.04182	-0.04841	-0.02200
Today	0.0301	-0.02616	-0.01025	-0.00245	-0.00690	-0.03486
	Volume	Today				
Year	0.5390	0.03010				
Lag1	0.0409	-0.02616				
Lag2	-0.0434	-0.01025				
Lag3	-0.0418	-0.00245				
Lag4	-0.0484	-0.00690				
Lag5	-0.0220	-0.03486				
Volume	1.0000	0.01459				
Today	0.0146	1.00000				

As one would expect, the correlations between the lag variables and today's returns are close to zero. In other words, there appears to be little correlation between today's returns and previous days' returns. The only substantial correlation is between **Year** and **Volume**. By plotting the data, which is ordered chronologically, we see that **Volume** is increasing over time. In other words, the average number of shares traded daily increased from 2001 to 2005.

```
> attach(Smarket)
> plot(Volume)
```

4.7.2 Logistic Regression

Next, we will fit a logistic regression model in order to predict **Direction** using **Lag1** through **Lag5** and **Volume**. The `glm()` function can be used to fit many types of generalized linear models, including logistic regression. The syntax of the `glm()` function is similar to that of `lm()`, except that we must pass in the argument `family = binomial` in order to tell **R** to run a logistic regression rather than some other type of generalized linear model.

`glm()`
generalized
linear model

```
> glm.fits <- glm(
  Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
  data = Smarket, family = binomial
)
> summary(glm.fits)
```

Call:

```
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5
    + Volume, family = binomial, data = Smarket)
```

Deviance Residuals:

```
Min      1Q  Median      3Q      Max
```

```

-1.45    -1.20     1.07     1.15     1.33

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.12600    0.24074  -0.52   0.60
Lag1        -0.07307    0.05017  -1.46   0.15
Lag2        -0.04230    0.05009  -0.84   0.40
Lag3         0.01109    0.04994   0.22   0.82
Lag4         0.00936    0.04997   0.19   0.85
Lag5         0.01031    0.04951   0.21   0.83
Volume       0.13544    0.15836   0.86   0.39

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1731.2 on 1249 degrees of freedom
Residual deviance: 1727.6 on 1243 degrees of freedom
AIC: 1742

Number of Fisher Scoring iterations: 3

```

The smallest p -value here is associated with **Lag1**. The negative coefficient for this predictor suggests that if the market had a positive return yesterday, then it is less likely to go up today. However, at a value of 0.15, the p -value is still relatively large, and so there is no clear evidence of a real association between **Lag1** and **Direction**.

We use the `coef()` function in order to access just the coefficients for this fitted model. We can also use the `summary()` function to access particular aspects of the fitted model, such as the p -values for the coefficients.

```

> coef(glm.fits)
(Intercept)      Lag1      Lag2      Lag3      Lag4
-0.12600    -0.07307    -0.04230     0.01109     0.00936
      Lag5      Volume
      0.01031     0.13544

> summary(glm.fits)$coef
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.12600    0.2407  -0.523   0.601
Lag1        -0.07307    0.0502  -1.457   0.145
Lag2        -0.04230    0.0501  -0.845   0.398
Lag3         0.01109    0.0499   0.222   0.824
Lag4         0.00936    0.0500   0.187   0.851
Lag5         0.01031    0.0495   0.208   0.835
Volume       0.13544    0.1584   0.855   0.392

> summary(glm.fits)$coef[, 4]
(Intercept)      Lag1      Lag2      Lag3      Lag4
      0.601      0.145      0.398      0.824      0.851
      Lag5      Volume
      0.835      0.392

```

The `predict()` function can be used to predict the probability that the market will go up, given values of the predictors. The `type = "response"` option tells R to output probabilities of the form $P(Y = 1|X)$, as opposed to other information such as the logit. If no data set is supplied to the

`predict()` function, then the probabilities are computed for the training data that was used to fit the logistic regression model. Here we have printed only the first ten probabilities. We know that these values correspond to the probability of the market going up, rather than down, because the `contrasts()` function indicates that **R** has created a dummy variable with a 1 for `Up`.

```
> glm.probs <- predict(glm.fits, type = "response")
> glm.probs[1:10]
  1      2      3      4      5      6      7      8      9     10
0.507 0.481 0.481 0.515 0.511 0.507 0.493 0.509 0.518 0.489
> contrasts(Direction)
      Up
Down   0
Up     1
```

In order to make a prediction as to whether the market will go up or down on a particular day, we must convert these predicted probabilities into class labels, `Up` or `Down`. The following two commands create a vector of class predictions based on whether the predicted probability of a market increase is greater than or less than 0.5.

```
> glm.pred <- rep("Down", 1250)
> glm.pred[glm.probs > .5] = "Up"
```

The first command creates a vector of 1,250 `Down` elements. The second line transforms to `Up` all of the elements for which the predicted probability of a market increase exceeds 0.5. Given these predictions, the `table()` function can be used to produce a confusion matrix in order to determine how many observations were correctly or incorrectly classified.

```
> table(glm.pred, Direction)
      Direction
glm.pred Down  Up
      Down  145 141
      Up    457 507
> (507 + 145) / 1250
[1] 0.5216
> mean(glm.pred == Direction)
[1] 0.5216
```

The diagonal elements of the confusion matrix indicate correct predictions, while the off-diagonals represent incorrect predictions. Hence our model correctly predicted that the market would go up on 507 days and that it would go down on 145 days, for a total of $507 + 145 = 652$ correct predictions. The `mean()` function can be used to compute the fraction of days for which the prediction was correct. In this case, logistic regression correctly predicted the movement of the market 52.2% of the time.

At first glance, it appears that the logistic regression model is working a little better than random guessing. However, this result is misleading because we trained and tested the model on the same set of 1,250 observations. In other words, $100\% - 52.2\% = 47.8\%$, is the *training* error

rate. As we have seen previously, the training error rate is often overly optimistic—it tends to underestimate the test error rate. In order to better assess the accuracy of the logistic regression model in this setting, we can fit the model using part of the data, and then examine how well it predicts the *held out* data. This will yield a more realistic error rate, in the sense that in practice we will be interested in our model's performance not on the data that we used to fit the model, but rather on days in the future for which the market's movements are unknown.

To implement this strategy, we will first create a vector corresponding to the observations from 2001 through 2004. We will then use this vector to create a held out data set of observations from 2005.

```
> train <- (Year < 2005)
> Smarket.2005 <- Smarket[!train, ]
> dim(Smarket.2005)
[1] 252    9
> Direction.2005 <- Direction[!train]
```

The object `train` is a vector of 1,250 elements, corresponding to the observations in our data set. The elements of the vector that correspond to observations that occurred before 2005 are set to `TRUE`, whereas those that correspond to observations in 2005 are set to `FALSE`. The object `train` is a *Boolean* vector, since its elements are `TRUE` and `FALSE`. Boolean vectors can be used to obtain a subset of the rows or columns of a matrix. For instance, the command `Smarket[train,]` would pick out a submatrix of the stock market data set, corresponding only to the dates before 2005, since those are the ones for which the elements of `train` are `TRUE`. The `!` symbol can be used to reverse all of the elements of a Boolean vector. That is, `!train` is a vector similar to `train`, except that the elements that are `TRUE` in `train` get swapped to `FALSE` in `!train`, and the elements that are `FALSE` in `train` get swapped to `TRUE` in `!train`. Therefore, `Smarket[!train,]` yields a submatrix of the stock market data containing only the observations for which `train` is `FALSE`—that is, the observations with dates in 2005. The output above indicates that there are 252 such observations.

boolean

We now fit a logistic regression model using only the subset of the observations that correspond to dates before 2005, using the `subset` argument. We then obtain predicted probabilities of the stock market going up for each of the days in our test set—that is, for the days in 2005.

```
> glm.fits <- glm(
  Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
  data = Smarket, family = binomial, subset = train
)
> glm.probs <- predict(glm.fits, Smarket.2005,
  type = "response")
```

Notice that we have trained and tested our model on two completely separate data sets: training was performed using only the dates before 2005,

and testing was performed using only the dates in 2005. Finally, we compute the predictions for 2005 and compare them to the actual movements of the market over that time period.

```
> glm.pred <- rep("Down", 252)
> glm.pred[glm.probs > .5] <- "Up"
> table(glm.pred, Direction.2005)
      Direction.2005
glm.pred Down Up
      Down   77 97
      Up    34 44
> mean(glm.pred == Direction.2005)
[1] 0.48
> mean(glm.pred != Direction.2005)
[1] 0.52
```

The `!=` notation means *not equal to*, and so the last command computes the test set error rate. The results are rather disappointing: the test error rate is 52%, which is worse than random guessing! Of course this result is not all that surprising, given that one would not generally expect to be able to use previous days' returns to predict future market performance. (After all, if it were possible to do so, then the authors of this book would be out striking it rich rather than writing a statistics textbook.)

We recall that the logistic regression model had very underwhelming *p*-values associated with all of the predictors, and that the smallest *p*-value, though not very small, corresponded to `Lag1`. Perhaps by removing the variables that appear not to be helpful in predicting `Direction`, we can obtain a more effective model. After all, using predictors that have no relationship with the response tends to cause a deterioration in the test error rate (since such predictors cause an increase in variance without a corresponding decrease in bias), and so removing such predictors may in turn yield an improvement. Below we have refit the logistic regression using just `Lag1` and `Lag2`, which seemed to have the highest predictive power in the original logistic regression model.

```
> glm.fits <- glm(Direction ~ Lag1 + Lag2, data = Smarket,
  family = binomial, subset = train)
> glm.probs <- predict(glm.fits, Smarket.2005,
  type = "response")
> glm.pred <- rep("Down", 252)
> glm.pred[glm.probs > .5] <- "Up"
> table(glm.pred, Direction.2005)
      Direction.2005
glm.pred Down Up
      Down   35 35
      Up    76 106
> mean(glm.pred == Direction.2005)
[1] 0.56
> 106 / (106 + 76)
[1] 0.582
```

Now the results appear to be a little better: 56% of the daily movements have been correctly predicted. It is worth noting that in this case, a much simpler strategy of predicting that the market will increase every day will also be correct 56% of the time! Hence, in terms of overall error rate, the logistic regression method is no better than the naive approach. However, the confusion matrix shows that on days when logistic regression predicts an increase in the market, it has a 58% accuracy rate. This suggests a possible trading strategy of buying on days when the model predicts an increasing market, and avoiding trades on days when a decrease is predicted. Of course one would need to investigate more carefully whether this small improvement was real or just due to random chance.

Suppose that we want to predict the returns associated with particular values of `Lag1` and `Lag2`. In particular, we want to predict `Direction` on a day when `Lag1` and `Lag2` equal 1.2 and 1.1, respectively, and on a day when they equal 1.5 and -0.8 . We do this using the `predict()` function.

```
> predict(glm.fits,
  newdata =
    data.frame(Lag1 = c(1.2, 1.5), Lag2 = c(1.1, -0.8)),
  type = "response"
)
      1      2
0.4791 0.4961
```