Tutorial 6

Quentin Bouet

2024-08-28

LDA, QDA, Naive Bayes and KNN

Suppose an anesthetist needs to determine whether an anesthetic is safe for a patient who is having a heart operation. The anesthesiologist may know certain things about this patient such as their age, gender, race, blood pressure and weight. Based on these kinds of data, the anesthesiologist would like to identify if the patient is safe or unsafe for anesthetic. This is a classification problem, we can employ one of many Classification methods for this. Last week, we started on logistic regression, this week we explore the use of Linear discriminant analysis (LDA), quadratic discriminant analysis (QDA) and K nearest approach (KNN).

Both LDA and QDA are based on the Gaussian mixture model framework, which assumes that the data is a mixture of K normal distributions, where K is the number of components. The probability of xi belong to component k is,

$$\Pr(y = k|x_i) = \frac{\pi_k f_k(x_i)}{\sum_{l=1}^{K} \pi_l f_l(x_i)}$$

where π_k is the "weight" of the component and fk() is a Gaussian distribution with mean of μ_k and variance σ_k.

The key difference between LDA and QDA is in the variance term. In LDA, it assumes variance is the same for all classes (or components) while QDA allows covariance matrix to vary between classes (i.e. k). This makes the boundaries (discriminant function) between classes more flexible (in quadratic form).

The discriminant function for LDA is

$$\hat{\delta}_k(x) = x\frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$$

and QDA

$$\delta_k(x) = -\frac{1}{2}x^T\mathbf{\Sigma}_k^{-1}x + x^T\mathbf{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\mathbf{\Sigma}_k^{-1}\mu_k + \log \pi_k$$

Here is just a snapshot of what we covered in the lecture. Now, it's your turn to work through the questions below in Group Two.

Discuss the following topics:

- Explain how LDA, QDA, Naive Bayes and KNN work

These are classification methods.

LDA and QDA use Gaussian Maximum Likelihood Classification, where they look at the probability of an observation being in one class or another (each class has a Multivariate Gaussian (Normal) Distribution). For LDA it is assumes that all classes have the same covariance.

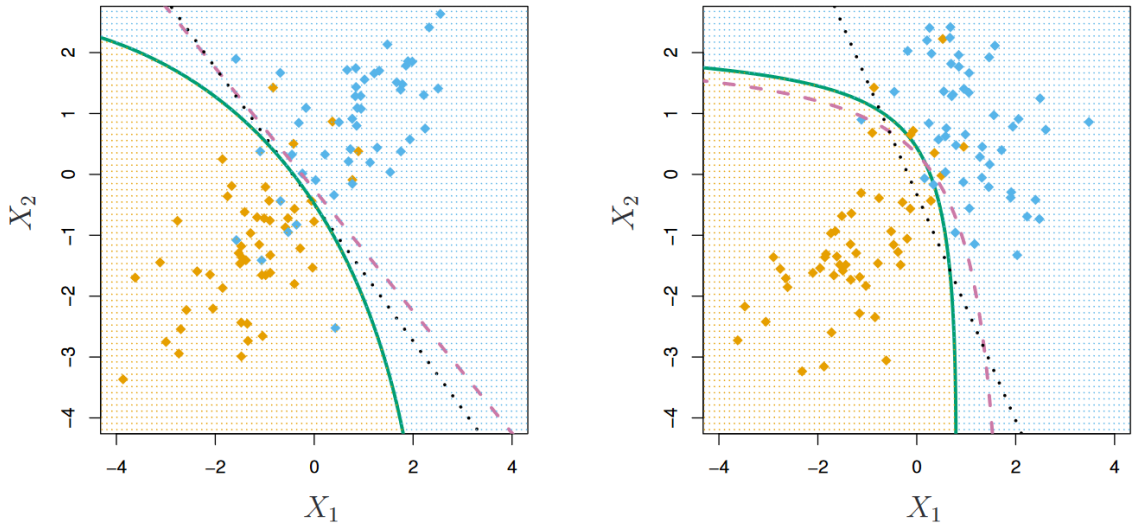
https://www.youtube.com/watch?v=IMfLXEOksGc

- What are the Key assumption behind LDA?

It assumes all classes have the same covariance matrix ($\sum_1 = \sum_2 = ... = \sum_k$).

- What are the key difference between LDA and QDA?

LDA assumes all classes have the same covariance matrix ($\sum_1 = \sum_2$) while QDA uses class-specific covariance matrices ($\sum_1 \neq \sum_2$). The decision boundaries between classes are linear for LDA, but quadratic (curved) for QDA:



left: $\sum_1 = \sum_2$

right: $\sum_1 \neq \sum_2$

where:

- Green - QDA

- black dotted - LDA

- purple - Naive Bayes

- Why does QDA use class-specific covariance matrices?

QDA uses class-specific covariance matrices to model the variability of data within each class more accurately. This flexibility is useful when the assumption of equal covariance matrices (as in LDA) is not valid.

- When would LDA be a better choice than QDA?

When the assumption of covariance matrices being equal ($\sum_1 = \sum_2$) is valid.

- What challenges might arise when using QDA with high-dimensional data?

QDA is computationally expensive when K (no of classes) and P (no of predictors) are large.

- What does k represent in K-nearest neighbor (KNN)?

"K" is not number of classes, but the number of "neighboring points" to be considered.

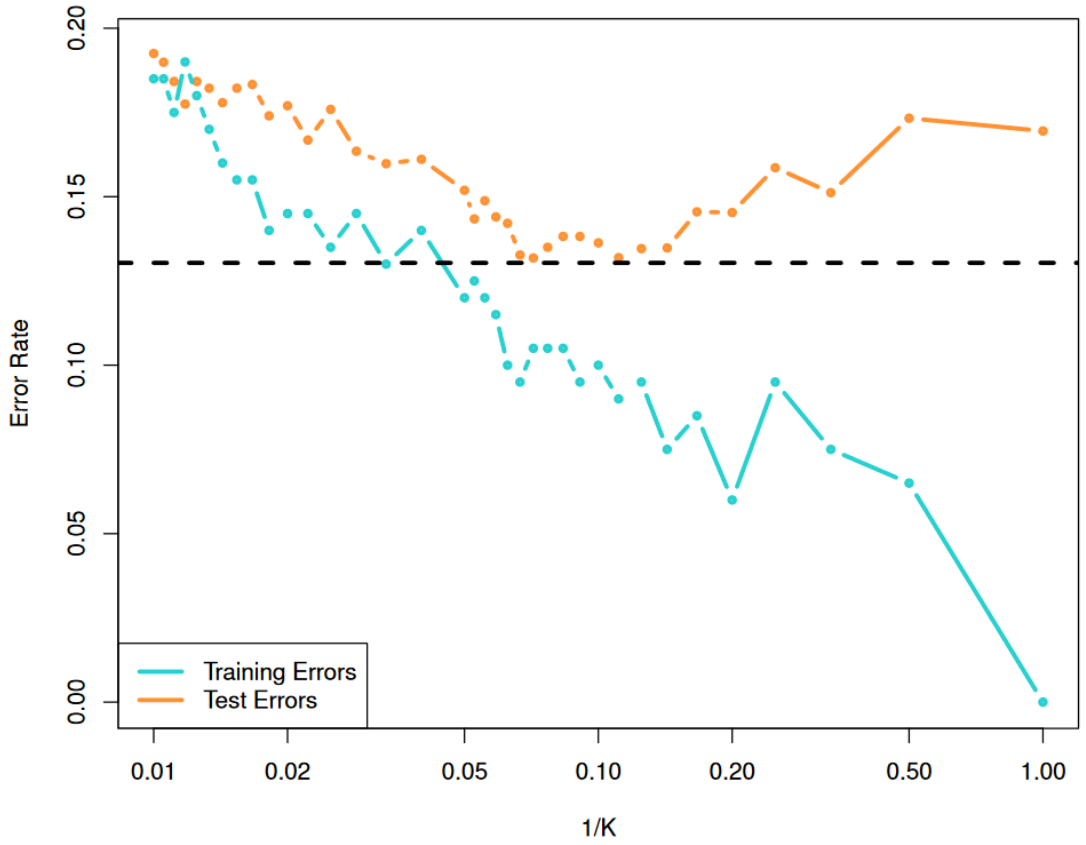
- How does the choice of k affect the performance of the KNN algorithm?

- A small K (e.g. K=1) is too flexible resulting in low bias but high variance.
- As K increases, bias increases but variance decreases.

Trade-off between BIAS and VARIANCE.

- How would you choose an optimal value for k in KNN?

1. Split data into training and test sets
2. Use test set to find the optimal K



- Can KNN be used for regression tasks, and if so, how?

Yes, KNN be used for regression tasks. It is like a moving average, where it takes the average of the K closest obeservations. This means that the average of the those obeservations will serve for prediction.

Labs - 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)
```

```
## Warning: package 'ISLR2' was built under R version 4.3.3
```

```
names(Smarket)
```

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

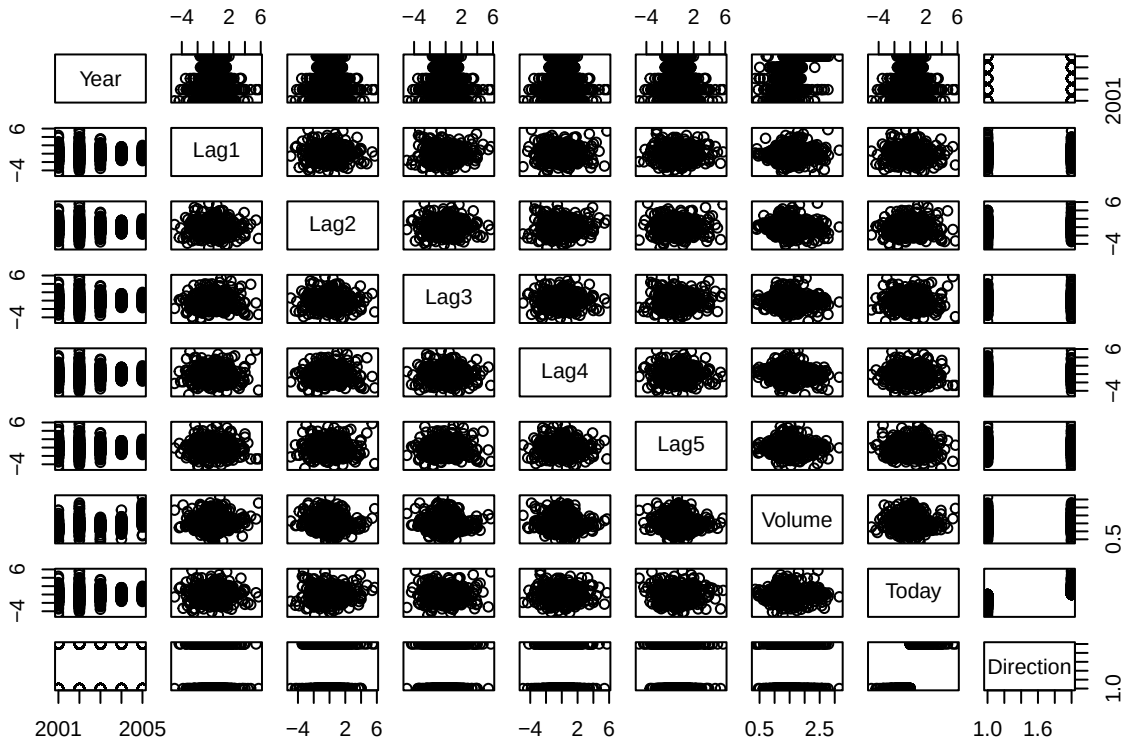
```
dim(Smarket)
```

```
## [1] 1250    9
```

```
summary(Smarket)
```

```
##       Year           Lag1                Lag2                Lag3
##  Min.   :2001   Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.922000
##  1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500   1st Qu.:-0.640000
##  Median :2003   Median : 0.039000   Median : 0.039000   Median : 0.038500
##  Mean   :2003   Mean   : 0.003834   Mean   : 0.003919   Mean   : 0.001716
##  3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.596750
##  Max.   :2005   Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.733000
##       Lag4                Lag5              Volume           Today
##  Min.   :-4.922000   Min.   :-4.92200   Min.   :0.3561   Min.   :-4.922000
##  1st Qu.:-0.640000   1st Qu.:-0.64000   1st Qu.:1.2574   1st Qu.:-0.639500
##  Median : 0.038500   Median : 0.03850   Median :1.4229   Median : 0.038500
##  Mean   : 0.001636   Mean   : 0.00561   Mean   :1.4783   Mean   : 0.003138
##  3rd Qu.: 0.596750   3rd Qu.: 0.59700   3rd Qu.:1.6417   3rd Qu.: 0.596750
##  Max.   : 5.733000   Max.   : 5.73300   Max.   :3.1525   Max.   : 5.733000
##  Direction
##  Down:602
##  Up  :648
##
##
##
##
```

```
pairs(Smarket)
```



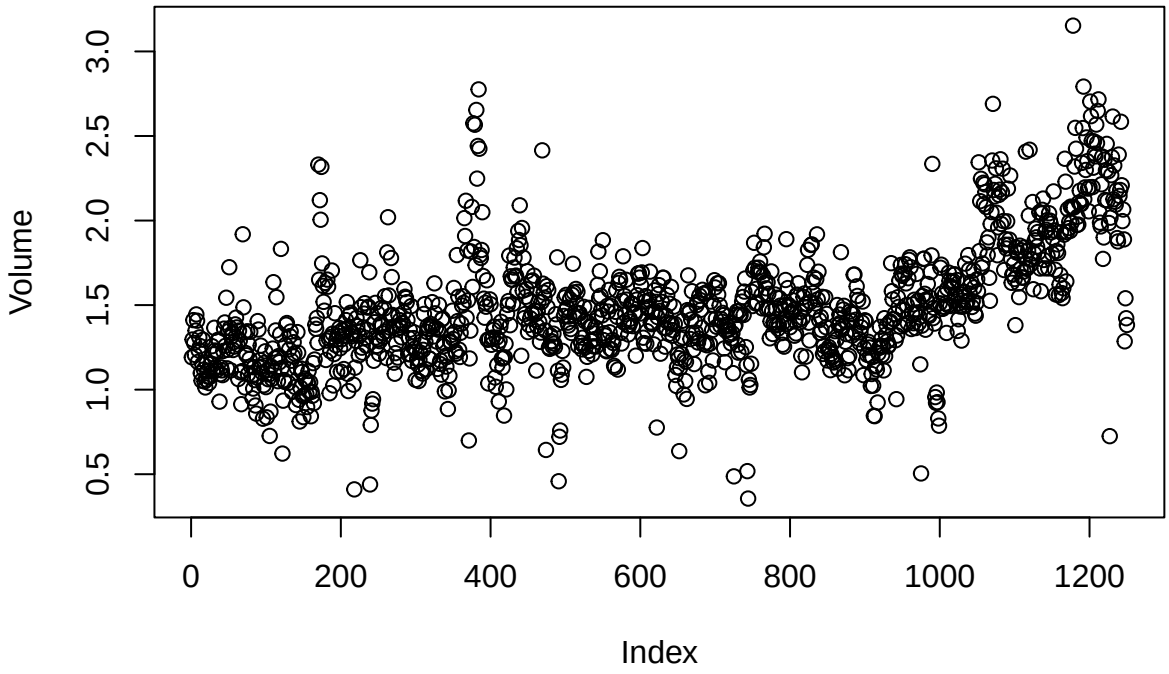
The cor() function produces a matrix that contains all of the pairwise correlations among the predictors in a data set. The first command below gives an error message because the Direction variable is qualitative.

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

```
##                Year         Lag1         Lag2         Lag3         Lag4
## Year    1.00000000  0.029699649  0.030596422  0.033194581  0.035688718
## Lag1    0.02969965  1.000000000 -0.026294328 -0.010803402 -0.002985911
## Lag2    0.03059642 -0.026294328  1.000000000 -0.025896670 -0.010853533
## Lag3    0.03319458 -0.010803402 -0.025896670  1.000000000 -0.024051036
## Lag4    0.03568872 -0.002985911 -0.010853533 -0.024051036  1.000000000
## Lag5    0.02978799 -0.005674606 -0.003557949 -0.018808338 -0.027083641
## Volume 0.53900647  0.040909908 -0.043383215 -0.041823686 -0.048414246
## Today  0.03009523 -0.026155045 -0.010250033 -0.002447647 -0.006899527
##                Lag5      Volume        Today
## Year    0.029787995  0.53900647  0.030095229
## Lag1   -0.005674606  0.04090991 -0.026155045
## Lag2   -0.003557949 -0.04338321 -0.010250033
## Lag3   -0.018808338 -0.04182369 -0.002447647
## Lag4   -0.027083641 -0.04841425 -0.006899527
## Lag5    1.000000000 -0.02200231 -0.034860083
## Volume -0.022002315  1.00000000  0.014591823
## Today  -0.034860083  0.01459182  1.000000000
```

As one would expect, the correlations between the lag variables and to- day'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 glm() many types of generalized linear models, including logistic regression. The generalized linear modelsyntax 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.fits <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Smarket , family = binomi
summary(glm.fits)
```

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

```
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.126000   0.240736  -0.523    0.601
## Lag1        -0.073074   0.050167  -1.457    0.145
## Lag2        -0.042301   0.050086  -0.845    0.398
## Lag3         0.011085   0.049939   0.222    0.824
## Lag4         0.009359   0.049974   0.187    0.851
## Lag5         0.010313   0.049511   0.208    0.835
## Volume       0.135441   0.158360   0.855    0.392
##
## (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: 1741.6
##
## 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         Lag5
## -0.126000257 -0.073073746 -0.042301344  0.011085108  0.009358938  0.010313068
##       Volume
##  0.135440659
```

```
summary(glm.fits)$coef
```

```
##                 Estimate Std. Error    z value  Pr(>|z|)
## (Intercept) -0.126000257 0.24073574 -0.5233966 0.6006983
## Lag1        -0.073073746 0.05016739 -1.4565986 0.1452272
## Lag2        -0.042301344 0.05008605 -0.8445733 0.3983491
## Lag3         0.011085108 0.04993854  0.2219750 0.8243333
## Lag4         0.009358938 0.04997413  0.1872757 0.8514445
## Lag5         0.010313068 0.04951146  0.2082966 0.8349974
## Volume       0.135440659 0.15835970  0.8552723 0.3924004
```

```
summary(glm.fits)$coef[, 4]
```

```
## (Intercept)        Lag1        Lag2        Lag3        Lag4        Lag5
##   0.6006983   0.1452272   0.3983491   0.8243333   0.8514445   0.8349974
##      Volume
##   0.3924004
```

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
## 0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565 0.4926509 0.5092292
##         9        10
## 0.5176135 0.4888378
```

```
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 table() 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 4.7 Lab: Classification Methods 175 because we trained and tested the model on the same set of 1, 250 ob- servations. 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 ob- servations 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 boolean 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 sub- matrix 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. We now fit a logistic regression model using only the subset of the obser- vations 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 sep- arate 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] 0.4801587
```

```
mean(glm.pred != Direction.2005)
```

```
## [1] 0.5198413
```

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] 0.5595238
```

```
106 / (106 + 76)
```

```
## [1] 0.5824176
```

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 4.7 Lab: Classification Methods 177 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 in- creasing 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
## 0.4791462 0.4960939
```

4.7.3 Linear Discriminant Analysis

Now we will perform LDA on the Smarket data. In R, we fit an LDA model using the lda() function, which is part of the MASS library. Notice that the lda() syntax for the lda() function is identical to that of lm(), and to that of glm() except for the absence of the family option. We fit the model using only the observations before 2005.

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 4.3.3
```

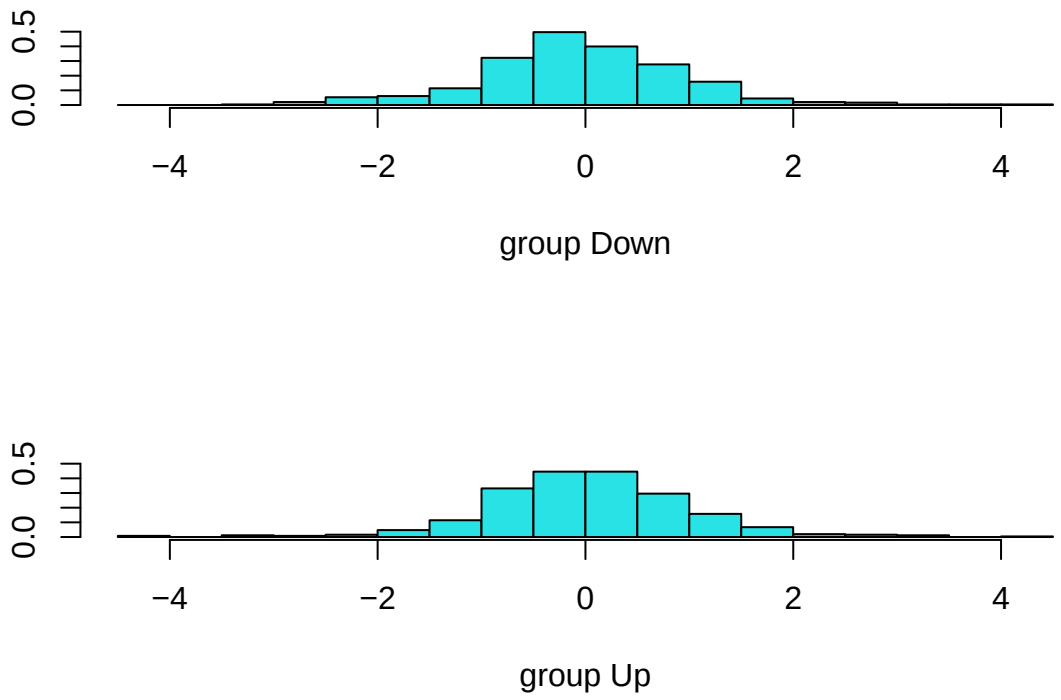
```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:ISLR2':
##
##     Boston
```

```
lda.fit <- lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
lda.fit
```

```
## Call:
## lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
##
## 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
```

```
plot(lda.fit)
```



The LDA output indicates that $\hat{\pi}_1 = 0.492$ and $\hat{\pi}_2 = 0.508$; in other words, 49.2 % of the training observations correspond to days during which the market went down. It also provides the group means; these are the average of each predictor within each class, and are used by LDA as estimates of μ_k. These suggest that there is a tendency for the previous 2 days' returns to be negative on days when the market increases, and a tendency for the previous days' returns to be positive on days when the market declines. The coefficients of linear discriminants output provides the linear combination of Lag1 and Lag2 that are used to form the LDA decision rule. In other words, these are the multipliers of the elements of X = x in (4.24). If -0.642 × Lag1-0.514 × Lag2 is large, then the LDA classifier will predict a market increase, and if it is small, then the LDA classifier will predict a market decline. The plot() function produces plots of the linear discriminants, obtained by computing -0.642 × Lag1-0.514 × Lag2 for each of the training obser- vations. The Up and Down observations are displayed separately. The predict() function returns a list with three elements. The first element, class, contains LDA's predictions about the movement of the mar- ket. The second element, posterior, is a matrix whose kth column contains the posterior probability that the corresponding observation belongs to the kth class, computed from (4.15). Finally, x contains the linear discrimi- nants, described earlier.

```
lda.pred <- predict(lda.fit , Smarket.2005)
names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

As we observed in Section 4.5, the LDA and logistic regression predictions are almost identical.

```
lda.class <- lda.pred$class
table(lda.class , Direction.2005)
```

```
##          Direction.2005
## lda.class Down  Up
##      Down   35  35
##      Up     76 106
```

```
mean(lda.class == Direction.2005)
```

```
## [1] 0.5595238
```

Applying a 50 % threshold to the posterior probabilities allows us to recreate the predictions contained in lda.pred$class.

```
sum(lda.pred$posterior[, 1] >= .5)
```

```
## [1] 70
```

```
sum(lda.pred$posterior[, 1] < .5)
```

```
## [1] 182
```

Notice that the posterior probability output by the model corresponds to the probability that the market will decrease:

```
lda.pred$posterior [1:20 , 1]
```

```
##       999      1000      1001      1002      1003      1004      1005      1006
## 0.4901792 0.4792185 0.4668185 0.4740011 0.4927877 0.4938562 0.4951016 0.4872861
##      1007      1008      1009      1010      1011      1012      1013      1014
## 0.4907013 0.4844026 0.4906963 0.5119988 0.4895152 0.4706761 0.4744593 0.4799583
##      1015      1016      1017      1018
## 0.4935775 0.5030894 0.4978806 0.4886331
```

```
lda.class [1:20]
```

```
##  [1] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Down Up   Up   Up
## [16] Up   Up   Down Up   Up
## Levels: Down Up
```

If we wanted to use a posterior probability threshold other than 50 % in order to make predictions, then we could easily do so. For instance, suppose that we wish to predict a market decrease only if we are very certain that the market will indeed decrease on that day—say, if the posterior probability is at least 90 %.

```
sum(lda.pred$posterior[, 1] > .9)
```

```
## [1] 0
```

No days in 2005 meet that threshold! In fact, the greatest posterior prob- ability of decrease in all of 2005 was 52.02 %.