Chapter 03 - Classification

March 27, 2022

1 Classification

In this lab we would be going through: - Logistic Regression - Linear Discriminant Analysis - Quadratic Discriminant Analysis

For this lab, we would examining the `Smarket` data set that contains a number of numeric variables plus a variable called `Direction` which has the two labels `Up` and `Down`.

Our goal is to predict `Direction` using the other features

[1]:
```
library(ISLR2)

#understanding Smarket data set
names(Smarket) #columns of the data set
dim(Smarket) #dimension of the data set
summary(Smarket)

# attach the data set to use the columns directly
attach(Smarket)
```

1. 'Year' 2. 'Lag1' 3. 'Lag2' 4. 'Lag3' 5. 'Lag4' 6. 'Lag5' 7. 'Volume' 8. 'Today' 9. 'Direction'

1. 1250 2. 9

```
      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
```

1.1 Logistic Regression

We are using the `glm()` function (it can be used to fit many types of generalized liner models) to fit a logistic regression model in order to predict `Direction` using `Lag1` - `Lag5` and `Volume`.

We need to pass in the argument `family = binomial` to `glm()` in order run logistic regression model rather than some other type of generalized linear model.

[2]:
```
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
-1.446   -1.203   1.065    1.145    1.326

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`(`0.15` although a relatively large) and the negative coefficient suggests that if the market had a positive return yesterday, then it is less likely to go up today.

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.

[3]:
```
glm.probs <- predict(glm.fits, type = "response")
glm.probs[1:10]
contrasts(Direction)
```

1 0.507084133395402 **2** 0.481467878454591 **3** 0.481138835214201 **4** 0.515222355813022 **5** 0.510781162691538 **6** 0.506956460534911 **7** 0.492650874187038 **8** 0.509229158207377 **9** 0.517613526170958 **10** 0.488837779771376

A matrix: 2 × 1 of type dbl

	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`

[4]:
```
glm.pred <- rep("Down", 1250)
glm.pred[glm.probs > .5] = "Up"

table(glm.pred, Direction)
```

```
        Direction
glm.pred Down  Up
    Down  145 141
    Up    457 507
```

[5]:
```
mean(glm.pred == Direction)
```

0.5216

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.

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.

[3]:
```
train <- (Year < 2005)

# Test data
Smarket.test <- Smarket[!train, ]
dim(Smarket.test)

#Train data
Smarket.train = Smarket[train, ]
dim(Smarket.train)

Direction.2005 = Direction[!train]
```

1. 252 2. 9

1. 998 2. 9

To fit the model using only the subset of the observations we can pass the `subset` argument to the `glm()` function along side other arguments

[38]:
```
#Return a logistic regression model over the training subset of data
#Response: Direction; Predictors: Lag1, Lag2
Smarket.train.fit = function(){
    # your code here
    return(glm(Direction ~ Lag1 + Lag2 , data = Smarket.train, family =␣
 ↪binomial))
}
fitted_train = Smarket.train.fit()
#Return the predicted probabilities object based on training fit over test data␣
 ↪set
Smarket.test.predict = function(){
    # your code here
    probs = predict(fitted_train, Smarket.test, type = "response")
    return(probs)
}
```

[39]:
```
fit = Smarket.train.fit()
coefficients = coef(fit)

#Test intercepts of the fit
stopifnot(round(coefficients['(Intercept)'],2)== 0.03)
stopifnot(round(coefficients['Lag1'],2)== -0.06)
stopifnot(round(coefficients['Lag2'],2)== -0.04)
```

[40]:
```
mean(Direction.2005)
```

```
Warning message in mean.default(Direction.2005):
"argument is not numeric or logical: returning NA"
```

<NA>

[42]:
```
predict = Smarket.test.predict()

glm.pred = rep('Down', 252)
glm.pred[predict > .5] = 'Up'

#Test mean of prediction
stopifnot(round(mean(glm.pred == Direction.2005), 2) == 0.56)
stopifnot(round(mean(glm.pred != Direction.2005), 2) == 0.44)
```

1.2 Linear Discriminant Analysis

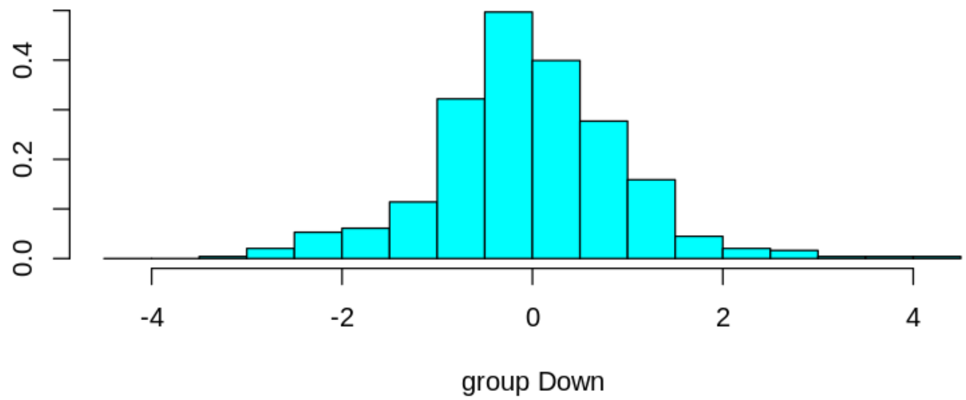
For this, we would be using `lda()` function which is a part of `MASS` library.

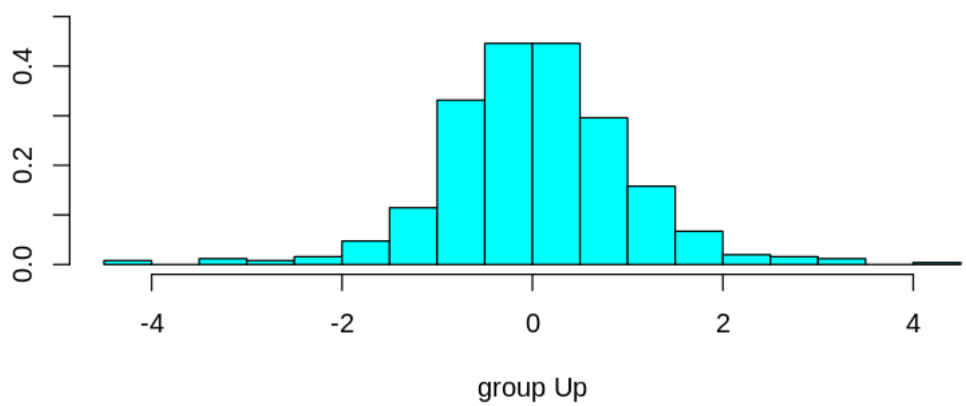
[43]:
```
library(MASS)
```

```
Attaching package: 'MASS'


The following object is masked from 'package:ISLR2':

    Boston
```

[44]:
```
lda.fit = lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
lda.fit

plot(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
```





The `plot()` function produces plots of the linear discriminants, obtained by computing $-0.642 \times$ `Lag1` $- 0.514 \times$ `Lag2` for each of the training obser- vations. The `Up` and `Down` observations are displayed separately

[45]:
```
#predict Direction based on the test data
lda.pred = predict(lda.fit, Smarket.test)
names(lda.pred)
```

1. 'class' 2. 'posterior' 3. 'x'

`predict()` function returns a list with three elements. - The first element, `class`, contains LDA's predictions about the movement of the market. - The second element, `posterior`, is a matrix whose `kth` column contains the posterior probability that the corresponding observation belongs to the `kth` class - Finally, `x` contains the linear discriminants, described earlier.

[46]:
```
lda.class = lda.pred$class
table(lda.class, Direction.2005)
mean(lda.class == Direction.2005)
```

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

0.55952380952381

Applying a 50 % threshold to the posterior probabilities allows us to recre- ate the predictions contained in `lda.pred$class`.

1.3 Quadratic Discriminant Analysis

The syntax of `qda()` is identical to that of an `lda()` The `predict()` function also works in the same fashion as for `lda()`

[56]:
```
#Return the fit using the training subset of data
#Response: Direction; Predictors: Lag1, Lag2
qda.fit = function(){
    # your code here
    return(qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train))
}
names(qda.fit()) #variables of the object
```

1. 'prior' 2. 'counts' 3. 'means' 4. 'scaling' 5. 'ldet' 6. 'lev' 7. 'N' 8. 'call' 9. 'terms' 10. 'xlevels'

[57]:
```
#Test - Count of predictions
stopifnot(qda.fit()$counts['Down'] == 491)
stopifnot(qda.fit()$counts['Up'] == 507)

#to understand more about qda() and output values
#?qda()
```

[59]:
```
qda.fit2 = qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
predict(qda.fit2, Smarket.test)$class
```

1. Up 2. Up 3. Up 4. Up 5. Up 6. Up 7. Up 8. Up 9. Up 10. Up 11. Up 12. Down 13. Up 14. Up 15. Up 16. Up 17. Up 18. Down 19. Up 20. Up 21. Up 22. Down 23. Down 24. Up 25. Down 26. Down 27. Up 28. Up 29. Up 30. Down 31. Up 32. Up 33. Up 34. Up 35. Up 36. Up 37. Up 38. Down 39. Down 40. Up 41. Up 42. Up 43. Up 44. Down 45. Down 46. Up 47. Up 48. Up 49. Up 50. Up 51. Up 52. Up 53. Up 54. Up 55. Up 56. Up 57. Up 58. Up 59. Up 60. Up 61. Down 62. Down 63. Up 64. Up 65. Up 66. Up 67. Up 68. Up 69. Up 70. Up 71. Up 72. Up 73. Up 74. Up 75. Down 76. Up 77. Down 78. Down 79. Up 80. Up 81. Up 82. Up 83. Up 84. Down 85. Up 86. Down 87. Down 88. Up 89. Up 90. Up 91. Up 92. Up 93. Up 94. Up 95. Down 96. Down 97. Down 98. Up 99. Up

100. Up 101. Up 102. Up 103. Up 104. Up 105. Up 106. Down 107. Up 108. Up 109. Up 110. Up 111. Up 112. Up 113. Up 114. Up 115. Up 116. Up 117. Up 118. Up 119. Up 120. Up 121. Up 122. Up 123. Up 124. Down 125. Up 126. Up 127. Up 128. Down 129. Up 130. Up 131. Down 132. Down 133. Up 134. Up 135. Up 136. Up 137. Up 138. Up 139. Down 140. Up 141. Up 142. Up 143. Up 144. Up 145. Down 146. Up 147. Up 148. Up 149. Up 150. Up 151. Up 152. Up 153. Up 154. Up 155. Up 156. Up 157. Up 158. Up 159. Up 160. Up 161. Up 162. Up 163. Up 164. Up 165. Up 166. Up 167. Up 168. Up 169. Up 170. Down 171. Up 172. Down 173. Down 174. Up 175. Up 176. Up 177. Up 178. Up 179. Up 180. Down 181. Up 182. Up 183. Up 184. Up 185. Up 186. Up 187. Up 188. Up 189. Down 190. Down 191. Up 192. Up 193. Up 194. Up 195. Up 196. Up 197. Up 198. Up 199. Up 200. Up 201. Down 202. Up 203. Down 204. Up 205. Up 206. Down 207. Down 208. Up 209. Up 210. Down 211. Down 212. Up 213. Up 214. Down 215. Up 216. Up 217. Up 218. Up 219. Down 220. Down 221. Up 222. Up 223. Up 224. Down 225. Down 226. Down 227. Down 228. Up 229. Up 230. Up 231. Up 232. Up 233. Up 234. Down 235. Up 236. Up 237. Up 238. Up 239. Up 240. Up 241. Up 242. Down 243. Up 244. Up 245. Up 246. Up 247. Up 248. Up 249. Up 250. Up 251. Up 252. Up

Levels: 1. 'Down' 2. 'Up'

[60]:
```
#Return the predicted class by fitting the data over Smarket.test
qda.predict.class = function(){
    # your code here
    qda_pred = predict(qda.fit2, Smarket.test)
    return(qda_pred$class)
}
table(qda.predict.class(), Direction.2005)
```

```
      Direction.2005
       Down  Up
  Down   30  20
  Up     81 121
```

[61]:
```
#Test the mean value of right predictions
stopifnot(round(mean(qda.predict.class() == Direction.2005),2) == 0.60)
```

[ ]:
```
```