FinalQuiz

March 28, 2022

1 Final Quiz

[1]:
```
library(ISLR2)
library(MASS)
library(e1071)
```

```
Attaching package: 'MASS'


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

    Boston
```

[2]:
```
attach(Auto)
head(Auto)
```

A data.frame: 6 × 9

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
	<dbl>	<int>	<dbl>	<int>	<int>	<dbl>	<int>	<int>
1	18	8	307	130	3504	12.0	70	1
2	15	8	350	165	3693	11.5	70	1
3	18	8	318	150	3436	11.0	70	1
4	16	8	304	150	3433	12.0	70	1
5	17	8	302	140	3449	10.5	70	1
6	15	8	429	198	4341	10.0	70	1

(a) Use the `lm()` function to perform a multiple linear regression with `mpg` as the response and all other variables except `name` as the predictors

[3]:
```
#lm.fit = ?
# your code here
lm.fit = lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration
 + year + origin, data=Auto)
summary(lm.fit)
```

```
Call:
```

```
lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
    acceleration + year + origin, data = Auto)

Residuals:
    Min      1Q  Median      3Q     Max
-9.5903 -2.1565 -0.1169  1.8690 13.0604

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  -17.218435   4.644294  -3.707  0.00024 ***
cylinders     -0.493376   0.323282  -1.526  0.12780
displacement   0.019896   0.007515   2.647  0.00844 **
horsepower    -0.016951   0.013787  -1.230  0.21963
weight        -0.006474   0.000652  -9.929  < 2e-16 ***
acceleration   0.080576   0.098845   0.815  0.41548
year           0.750773   0.050973  14.729  < 2e-16 ***
origin         1.426141   0.278136   5.127 4.67e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.328 on 384 degrees of freedom
Multiple R-squared:  0.8215,Adjusted R-squared:  0.8182
F-statistic: 252.4 on 7 and 384 DF,  p-value: < 2.2e-16
```

[4]:
```
#hidden test cases
```

(b) Add an interaction term between **horsepower** and **weight** to the set of existant predictors and define a fit

[5]:
```
#lm.fit = ?
# your code here
lm.fit = lm(lm(mpg ~ cylinders + displacement + horsepower + weight +
 ↪acceleration + year + origin + horsepower:weight
                 , data=Auto))
summary(lm.fit)
```

```
Call:
lm(formula = lm(mpg ~ cylinders + displacement + horsepower +
    weight + acceleration + year + origin + horsepower:weight,
    data = Auto))

Residuals:
   Min     1Q Median     3Q    Max
-8.589 -1.617 -0.184  1.541 12.001

Coefficients:
```

```
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        2.876e+00  4.511e+00   0.638 0.524147
cylinders         -2.955e-02  2.881e-01  -0.103 0.918363
displacement       5.950e-03  6.750e-03   0.881 0.378610
horsepower        -2.313e-01  2.363e-02  -9.791  < 2e-16 ***
weight            -1.121e-02  7.285e-04 -15.393  < 2e-16 ***
acceleration      -9.019e-02  8.855e-02  -1.019 0.309081
year               7.695e-01  4.494e-02  17.124  < 2e-16 ***
origin             8.344e-01  2.513e-01   3.320 0.000986 ***
horsepower:weight  5.529e-05  5.227e-06  10.577  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.931 on 383 degrees of freedom
Multiple R-squared:  0.8618,Adjusted R-squared:  0.859
F-statistic: 298.6 on 8 and 383 DF,  p-value: < 2.2e-16
```

[6]:

```
#hidden test cases
```

(c) Add another predictor that is the sqaure of `horsepower` to the existant set of predictors (including the interaction term in (b)) and define a fit

[7]:

```
#lm.fit = ?
# your code here
lm.fit = lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration
 + year + origin + horsepower:weight + horsepower^2
              , data=Auto)
summary(lm.fit)
```

```
Call:
lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
    acceleration + year + origin + horsepower:weight + horsepower^2,
    data = Auto)

Residuals:
   Min     1Q Median     3Q    Max
-8.589 -1.617 -0.184  1.541 12.001

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        2.876e+00  4.511e+00   0.638 0.524147
cylinders         -2.955e-02  2.881e-01  -0.103 0.918363
displacement       5.950e-03  6.750e-03   0.881 0.378610
horsepower        -2.313e-01  2.363e-02  -9.791  < 2e-16 ***
weight            -1.121e-02  7.285e-04 -15.393  < 2e-16 ***
acceleration      -9.019e-02  8.855e-02  -1.019 0.309081
```

```
year               7.695e-01  4.494e-02  17.124  < 2e-16 ***
origin             8.344e-01  2.513e-01   3.320 0.000986 ***
horsepower:weight  5.529e-05  5.227e-06  10.577  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.931 on 383 degrees of freedom
Multiple R-squared:  0.8618,Adjusted R-squared:  0.859
F-statistic: 298.6 on 8 and 383 DF,  p-value: < 2.2e-16
```

[8]:
```
#hidden test cases
```

(d) Create a binary variable, `mpg01`, that contains a `1` if `mpg` contains a value above its median, and a `0` if mpg contains a value below its median. You can compute the median using the `median()` function.

[9]:
```
#mpg01 = ?
# your code here
mpg01 = ifelse(Auto$mpg > median(mpg), 1, 0)
mpg01
```

1. 0 2. 0 3. 0 4. 0 5. 0 6. 0 7. 0 8. 0 9. 0 10. 0 11. 0 12. 0 13. 0 14. 0 15. 1 16. 0 17. 0 18. 0 19. 1 20. 1 21. 1 22. 1 23. 1 24. 1 25. 0 26. 0 27. 0 28. 0 29. 0 30. 1 31. 1 32. 1 33. 0 34. 0 35. 0 36. 0 37. 0 38. 0 39. 0 40. 0 41. 0 42. 0 43. 0 44. 0 45. 0 46. 0 47. 0 48. 0 49. 1 50. 1 51. 1 52. 1 53. 1 54. 1 55. 1 56. 1 57. 1 58. 1 59. 1 60. 0 61. 0 62. 0 63. 0 64. 0 65. 0 66. 0 67. 0 68. 0 69. 0 70. 0 71. 0 72. 0 73. 0 74. 0 75. 0 76. 0 77. 0 78. 0 79. 1 80. 0 81. 1 82. 1 83. 1 84. 1 85. 0 86. 0 87. 0 88. 0 89. 0 90. 0 91. 0 92. 0 93. 0 94. 0 95. 0 96. 0 97. 0 98. 0 99. 0 100. 0 101. 1 102. 1 103. 0 104. 0 105. 0 106. 0 107. 0 108. 0 109. 0 110. 0 111. 0 112. 0 113. 0 114. 1 115. 0 116. 0 117. 1 118. 1 119. 0 120. 0 121. 0 122. 1 123. 0 124. 0 125. 0 126. 0 127. 0 128. 1 129. 1 130. 1 131. 1 132. 0 133. 0 134. 0 135. 0 136. 0 137. 0 138. 0 139. 0 140. 1 141. 1 142. 1 143. 1 144. 1 145. 1 146. 1 147. 1 148. 1 149. 1 150. 1 151. 0 152. 0 153. 0 154. 0 155. 0 156. 0 157. 0 158. 0 159. 0 160. 0 161. 0 162. 0 163. 0 164. 0 165. 0 166. 1 167. 1 168. 0 169. 1 170. 1 171. 1 172. 1 173. 0 174. 1 175. 0 176. 1 177. 1 178. 0 179. 1 180. 1 181. 1 182. 1 183. 1 184. 1 185. 1 186. 0 187. 0 188. 0 189. 0 190. 0 191. 0 192. 1 193. 0 194. 1 195. 1 196. 1 197. 1 198. 0 199. 0 200. 0 201. 0 202. 1 203. 1 204. 1 205. 1 206. 0 207. 0 208. 0 209. 0 210. 0 211. 0 212. 0 213. 0 214. 0 215. 1 216. 1 217. 1 218. 1 219. 1 220. 0 221. 0 222. 0 223. 0 224. 0 225. 0 226. 0 227. 0 228. 0 229. 0 230. 0 231. 0 232. 1 233. 1 234. 1 235. 1 236. 1 237. 1 238. 1 239. 1 240. 0 241. 0 242. 0 243. 1 244. 1 245. 1 246. 1 247. 1 248. 0 249. 0 250. 0 251. 0 252. 0 253. 0 254. 1 255. 0 256. 0 257. 0 258. 0 259. 0 260. 0 261. 0 262. 0 263. 0 264. 0 265. 1 266. 1 267. 1 268. 1 269. 0 270. 1 271. 1 272. 1 273. 0 274. 0 275. 0 276. 0 277. 1 278. 1 279. 0 280. 0 281. 0 282. 0 283. 0 284. 0 285. 0 286. 0 287. 0 288. 0 289. 0 290. 0 291. 0 292. 1 293. 1 294. 1 295. 1 296. 1 297. 1 298. 1 299. 1 300. 1 301. 1 302. 1 303. 1 304. 1 305. 1 306. 1 307. 1 308. 1 309. 1 310. 1 311. 1 312. 1 313. 1 314. 1 315. 0 316. 1 317. 1 318. 1 319. 1 320. 1 321. 1 322. 1 323. 1 324. 1 325. 1 326. 1 327. 1 328. 1 329. 1 330. 1 331. 1 332. 1 333. 1 334. 1 335. 1 336. 1 337. 1 338. 1 339. 1 340. 1 341. 1 342. 1 343. 1 344. 1 345. 1 346. 1 347. 1 348. 1 349. 1 350. 1 351. 1 352. 1 353. 1 354. 1 355. 1 356. 1 357. 1 358. 1 359. 0 360. 1 361. 0 362. 0 363. 1 364. 1 365. 1 366. 1 367. 1 368. 1 369. 1 370. 1 371. 1 372. 1 373. 1 374. 1 375. 1 376. 1 377. 1 378. 1 379. 1 380. 1 381. 1 382. 1 383. 1 384. 0 385. 1 386. 1 387. 1 388. 1 389. 1 390. 1 391. 1 392. 1

```
[10]: #hidden test cases
```

```
[11]: #Adding mpg01 to the data frame and removing mpg
Auto$mpg01 = mpg01
Auto$mpg = NULL

#splitting data into training and test data sets
n = dim(Auto)[1]
inds.train = sample(1:n,3*n/4)
Auto.train = Auto[inds.train,]
inds.test = (1:n)[-inds.train]
Auto.test = Auto[inds.test,]
```

```
[12]: head(Auto.train)
```

A data.frame: 6 × 9

	cylinders <int>	displacement <dbl>	horsepower <int>	weight <int>	acceleration <dbl>	year <int>	origin <int>	name <fct
160	8	351	148	4657	13.5	75	1	ford
231	8	350	170	4165	11.4	77	1	chev
345	4	86	64	1875	16.4	81	1	plym
179	4	120	88	2957	17.0	75	2	peug
260	6	200	85	3070	16.7	78	1	merc
144	4	97	78	2300	14.5	74	2	opel

(e) Perform `LDA` on the training data in order to predict `mpg01` using the variables that seemed most associated (`cylinders`, `displacement`, `weight`) with `mpg01`. What is the test error of the model obtained(accuracy)?

```
[13]: #accuracy = ?
lda.fit = lda(mpg01 ~ cylinders + displacement + weight, data=Auto.train,␣
 ↪subset=inds.train)

lda.accuracy = function(model=lda.fit){

    # your code here
    lda.pred = predict(model, Auto.test)
    lda.class = lda.pred$class
    return (mean(lda.class == Auto.test$mpg01))
}

accuracy = lda.accuracy()
```

```
[14]: lda.accuracy()
```

0.877551020408163

```
[15]: #hidden test cases
```

(f) Perform `QDA` similar to (e) and obtain the accuracy of the model??

[16]:
```
#accuracy = ?

qda.fit = qda(mpg01 ~ cylinders + displacement + weight, data=Auto.train,␣
↪subset=inds.train)

qda.accuracy = function(model=qda.fit){

   # your code here
    qda.pred = predict(model, Auto.test)
    qda.class = qda.pred$class
    return (mean(qda.class == Auto.test$mpg01))
}

accuracy = qda.accuracy()
```

[17]:
```
qda.accuracy()
```

0.86734693877551

[18]:
```
#hidden test cases
```

(g) Perform NaiveBayes similar to (e) and obtain the accuracy of the model??

[19]:
```
#accuracy = ?
nb.fit = naiveBayes(mpg01 ~ cylinders + displacement + weight, data=Auto.train,␣
↪subset=inds.train)

nb.accuracy = function(model=nb.fit){
    # your code here
    nb.class = predict(model, Auto.test)

    return(mean(nb.class == Auto.test$mpg01))
}

accuracy = nb.accuracy()
```

[20]:
```
nb.accuracy()
```

0.86734693877551

[21]:
```
#hidden test cases
```

[ ]:
```

```