da410\_project3

Jason Grahn

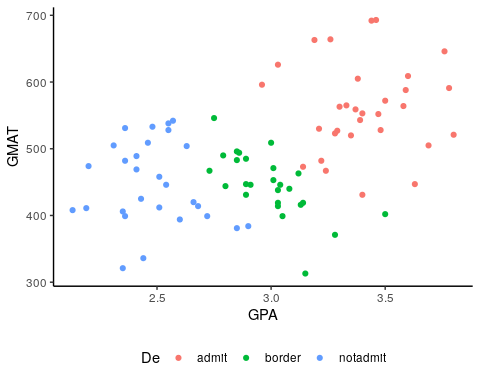
1/27/2019

# Problem 1.

## 1. Use admission.csv as a training dataset.

We’re going to use admission.csv twice, so importing once and calling it good.

admission <- readr::read\_csv(here::here("project3/admission.csv"),   
 col\_types = cols(De = col\_factor(levels = c("admit", "border", "notadmit"))))  
  
#plot scatter to look for clustering  
admission %>%   
 ggplot(aes(x = GPA, y = GMAT, color = De)) +  
 geom\_point() +  
 theme\_classic() +  
 theme(legend.position = "bottom")



## 2. Train model using LDA by setting admit/not-admit/border with the same probabilities.

model\_1 <- lda(formula = De ~ .,   
 data = admission,   
 prior = c(1,1,1)/3)  
  
model\_1

## Call:  
## lda(De ~ ., data = admission, prior = c(1, 1, 1)/3)  
##   
## Prior probabilities of groups:  
## admit border notadmit   
## 0.3333333 0.3333333 0.3333333   
##   
## Group means:  
## GPA GMAT  
## admit 3.403871 561.2258  
## border 2.992692 446.2308  
## notadmit 2.482500 447.0714  
##   
## Coefficients of linear discriminants:  
## LD1 LD2  
## GPA 5.017202736 1.85401003  
## GMAT 0.008503148 -0.01448967  
##   
## Proportion of trace:  
## LD1 LD2   
## 0.9644 0.0356

## 3. Calculate the misclassfication rate

#apply the model to admissions with predict()  
predict\_admit <- predict(model\_1, # predictions  
 data = admissions)  
  
#pull the predictions and  
predict\_class <- predict\_admit$class  
  
#I might want to use that data later, so let's write them back to the table  
admission\_predictions <- admission %>%   
 mutate(predictions = predict\_admit$class,  
 LD1 = predict\_admit$x[,1],  
 LD2 = predict\_admit$x[,2])  
  
#back to misclassification  
#quick glance of predictions versus actuals  
table(predict\_class, admission$De)

##   
## predict\_class admit border notadmit  
## admit 27 1 0  
## border 4 25 2  
## notadmit 0 0 26

first\_percentages <- admission\_predictions %>%   
 mutate(correct\_class = factor((De == predictions))) %>%   
 group\_by(correct\_class) %>%   
 summarise(classified\_count = n(),   
 total = nrow(admission\_predictions)) %>%   
 mutate(percentage = round(classified\_count / total \* 100, 2))   
  
first\_percentages

## # A tibble: 2 x 4  
## correct\_class classified\_count total percentage  
## <fct> <int> <int> <dbl>  
## 1 FALSE 7 85 8.24  
## 2 TRUE 78 85 91.8

## 4. Predict students with GPA and SAT score as below.

#the values  
GPA <- c(3.14, 3.08, 2.08, 3.22)  
GMAT <- c(470, 591, 641, 463)  
p\_student <- as.factor(c("student1", "student2", "student3", "student4"))  
  
#put those in a table  
predict\_me <- tibble(p\_student, GPA, GMAT)  
  
#apply the model and save the output  
predict\_subgroup <- predict(model\_1, # predictions  
 newdata = predict\_me)  
  
#incorporate the output into the original table  
predict\_me\_output <- predict\_me %>%   
 mutate(predictions = predict\_subgroup$class)  
  
predict\_me\_output

## # A tibble: 4 x 4  
## p\_student GPA GMAT predictions  
## <fct> <dbl> <dbl> <fct>   
## 1 student1 3.14 470 border   
## 2 student2 3.08 591 admit   
## 3 student3 2.08 641 notadmit   
## 4 student4 3.22 463 border

# Problem 2.

## 1. Use admission.csv as a training dataset.

We already input this before, so let’s call it again just to show we have it.

head(admission,5)

## # A tibble: 5 x 3  
## GPA GMAT De   
## <dbl> <dbl> <fct>  
## 1 2.96 596 admit  
## 2 3.14 473 admit  
## 3 3.22 482 admit  
## 4 3.29 527 admit  
## 5 3.69 505 admit

## 2. Train model using LDA by setting probability of admit is 50% while probability of not admit is 25% and probability of border is 25%.

model\_2 <- lda(formula = De ~ .,   
 data = admission,   
 prior = c(.5,.25,.25))  
  
model\_2

## Call:  
## lda(De ~ ., data = admission, prior = c(0.5, 0.25, 0.25))  
##   
## Prior probabilities of groups:  
## admit border notadmit   
## 0.50 0.25 0.25   
##   
## Group means:  
## GPA GMAT  
## admit 3.403871 561.2258  
## border 2.992692 446.2308  
## notadmit 2.482500 447.0714  
##   
## Coefficients of linear discriminants:  
## LD1 LD2  
## GPA 4.961868967 1.9973815  
## GMAT 0.008915905 -0.0142394  
##   
## Proportion of trace:  
## LD1 LD2   
## 0.9724 0.0276

## 3. Calculate the misclassfication rate

#apply the model to admissions with predict()  
predict\_admit\_2 <- predict(model\_2, # predictions  
 data = admissions)  
  
#pull the classificaton predictions into their own frame  
predict\_class\_2 <- predict\_admit\_2$class  
  
#I might want to use that data later, so let's write these back to the table as the other predictions  
admission\_predictions <- admission\_predictions %>%   
 mutate(predictions\_2 = predict\_admit\_2$class,  
 LD1\_2 = predict\_admit\_2$x[,1],  
 LD2\_2 = predict\_admit\_2$x[,2])  
  
#back to misclassification  
#quick glance of predictions versus actuals  
table(predict\_class\_2, admission$De)

##   
## predict\_class\_2 admit border notadmit  
## admit 29 1 0  
## border 2 25 2  
## notadmit 0 0 26

second\_percentages <- admission\_predictions %>%   
 mutate(correct\_class\_2 = factor((De == predictions\_2))) %>%   
 group\_by(correct\_class\_2) %>%   
 summarise(classified\_count = n(),   
 total = nrow(admission\_predictions)) %>%   
 mutate(percentage = round(classified\_count / total \* 100, 2))

## 4. Predict students with GPA and SAT score as below.

#apply the model and save the output  
predict\_subgroup\_2 <- predict(model\_2, # predictions  
 newdata = predict\_me)  
  
#incorporate the output into the previous prediction table  
predict\_me\_output <- predict\_me\_output %>%   
 mutate(predictions\_2 = predict\_subgroup\_2$class)  
  
predict\_me\_output

## # A tibble: 4 x 5  
## p\_student GPA GMAT predictions predictions\_2  
## <fct> <dbl> <dbl> <fct> <fct>   
## 1 student1 3.14 470 border border   
## 2 student2 3.08 591 admit admit   
## 3 student3 2.08 641 notadmit notadmit   
## 4 student4 3.22 463 border border

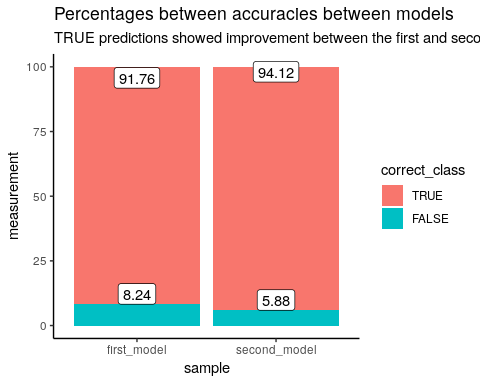
## Compare differences of the result from problem1.

#One way to compare results is to look at the difference in percentages between missclassfied students  
  
# a negative number indicates that the second model was worse than the first.   
# ie; the second model incorrectly predicted a larger count than the first.  
  
#a positive number indicates that the second model was better, that it more accurately predicted the actual  
first\_percentages$percentage[1] - second\_percentages$percentage[1]

## [1] 2.36

We see that the second model decreased the misclassification rate by 2.36 points; that it accurately predicted the actual better than the first model.

first\_percentages %>%   
 dplyr::select(correct\_class, percentage) %>%   
 rename(first\_model = percentage) %>%   
 mutate(second\_model = (second\_percentages$percentage),  
 correct\_class = factor(correct\_class, levels = c(TRUE, FALSE))) %>%   
 gather(key = sample, value = measurement, -correct\_class) %>%   
 ggplot(aes(x = sample, y = measurement, label = measurement)) +  
 # scale\_x\_discrete(position = "top") +  
 geom\_col(aes(fill = correct\_class)) +  
 theme\_classic() +   
 geom\_label(vjust = "bottom") +  
 labs(title = "Percentages between accuracies between models",  
 subtitle = "TRUE predictions showed improvement between the first and second model.")



We should check for equal covariance just to make sure we’re good.

heplots::boxM(cbind(GPA, GMAT) ~ De,   
 data = admission)

##   
## Box's M-test for Homogeneity of Covariance Matrices  
##   
## data: Y  
## Chi-Sq (approx.) = 16.074, df = 6, p-value = 0.01336

With a p-value of 0.0133598 we can be assured that the test of homogeneity of covariance passes, that the values are not covariant.

# Problem 3.

## Explain what is Quadratic Discriminant Analysis (QDA), and use QDA to train the model, discuss if this project can be done better by QDA, why or why not.

Where LDA utilizes a linear boundary (straight lines) to separate classifiers, QDA utilizes non-linear boundaries. QDA fitment results in curved fitments (often multiple times over). This is excellent for classifications that have numerous variables and large data sets, but in small sample sizes results in poor performance due to overfitting.

model\_qda <- MASS::qda(formula = De ~ .,   
 data = admission,   
 prior = c(.5,.25,.25))  
#apply the QDA model to admissions with predict()  
predict\_qda <- predict(model\_qda, # predictions  
 data = admissions)  
  
#pull the classificaton predictions into their own frame  
predict\_qda\_output <- predict\_qda$class  
  
#I might want to use that data later, so let's write these back to the table as the other predictions  
admission\_predictions <- admission\_predictions %>%   
 mutate(predictions\_qda = predict\_qda\_output)  
  
#back to misclassification  
#quick glance of predictions versus actuals  
#table(predict\_qda\_output, admission$De)  
  
third\_percentages <- admission\_predictions %>%   
 mutate(correct\_class\_3 = factor((De == predictions\_qda))) %>%   
 group\_by(correct\_class\_3) %>%   
 summarise(classified\_count = n(),   
 total = nrow(admission\_predictions)) %>%   
 mutate(percentage = round(classified\_count / total \* 100, 2))   
  
knitr::kable(third\_percentages) %>%  
 kableExtra::kable\_styling(bootstrap\_options = "striped")

## Warning in kableExtra::kable\_styling(., bootstrap\_options = "striped"):  
## Please specify format in kable. kableExtra can customize either HTML or  
## LaTeX outputs. See https://haozhu233.github.io/kableExtra/ for details.

|  |  |  |  |
| --- | --- | --- | --- |
| correct\_class\_3 | classified\_count | total | percentage |
| FALSE | 3 | 85 | 3.53 |
| TRUE | 82 | 85 | 96.47 |

I believe that LDA (method 2) is the better method to use for this project. While it does have a more accurate classification rate, given the sample size and quantity of variables, QDA is potentially over fitting the model to the data.