ChessLDA

Leonard Orozco

2020/3/12

```
library(MASS)
games <- read.csv('games_new_vars.csv')
attach(games)
```

In this episode of the Leo data science series, we will hit our chess data set with a Linear Discriminant Analysis. Check out our notebook for more info on specifics.

```
ld.fit <- lda(result ~ abs_diff_rating, data = games)
ld.fit
```

```
## Call:
## lda(result ~ abs_diff_rating, data = games)
##
## Prior probabilities of groups:
##          0          1          2
## 0.33682321 0.04736265 0.61581414
##
## Group means:
##   abs_diff_rating
## 0        116.0524
## 1        147.6284
## 2        206.2477
##
## Coefficients of linear discriminants:
##                          LD1
## abs_diff_rating 0.005743204
```

Our prior probabilities correspond to the proportion of observations that fall in each class. Our probabilities should all add up to one.

```
ld.fit$prior[1] + ld.fit$prior[2] + ld.fit$prior[3]
```

```
## 0
## 1
```

Which they do.

Our group means suggest that when we see a loss in the higher rated player, on average the absolute difference in rating is 116.0524. Likewise, when draws occur, the average absolute difference in rating is 147.6284. Finally, if we see a win from the higher rated player, we expect the absolute difference in rating to be on average 206.2477.

The results for draws surprises. I would expect a small absolute difference in rating to have more draws. These results are also quite the sham since I trained the model on the entire set.

This is all about familiarization so we will do a simple cross set validation with a split of 70 / 30 for the data. For this, we will refit on training before testing as well.

```
n = nrow(games)
shuffle.data <-   games[sample(n), ]
train.data <-  shuffle.data[1:round(.7*n), ]
test.data <-  shuffle.data[(round(.7*n)+1):n, ]
```

```
lda.fit <- lda(result ~ abs_diff_rating, data = train.data)
lda.fit
```

```
## Call:
## lda(result ~ abs_diff_rating, data = train.data)
##
## Prior probabilities of groups:
##          0          1          2
## 0.33672815 0.04828716 0.61498469
##
## Group means:
##   abs_diff_rating
## 0        117.4769
## 1        148.7684
## 2        206.5357
##
## Coefficients of linear discriminants:
##                           LD1
## abs_diff_rating 0.005752557
```

Similar results to our first model.

```
predictions <- predict(lda.fit, test.data)
table(predictions$class)
```

```
##
##    0    1    2
##    0    0 6017
```

We predicted a win for the higher rating for everyone it seems. Uh oh. Big yike houston.

It could be that our test.data really did have a win win for everyone (hah!). Let's check.

```
# I use a constant 2 since our predictions were all wins.
mistakes <- 2 - test.data$result
sum(mistakes) / nrow(test.data)
```

```
## [1] 0.7192953
```

The error rate for this model is stunningly awful.

Which is almost surprising. We can check the posterior probabilities (simplified, these probabilities dictate what class we will predict for the observation, if their probability of being in a class is higher than 50%, we will predict that class for the observation).

```
head(predictions$posterior, 5)
```

```
##               0          1         2
## 7379  0.2828001 0.04581866 0.6713813
## 6292  0.2879834 0.04622564 0.6657909
## 17481 0.4238957 0.05451758 0.5215867
## 8581  0.2200474 0.04028514 0.7396675
## 8481  0.2742802 0.04513391 0.6805859
```

Take a look at column 2 (i.e. probabilities of being a win for the higher rating). Your eyes do not lie. For these observations, there is at least a 50% of winning for the higher rating. In fact, every posterior probability for column 2 is at least 50%.

```
sum(predictions$posterior[,3] < 0.5)
```

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

Hence our model will give every observation a win classification. Bad model. We'll have to spank it into being good.

Despite my current prowess in R, I haven't yet figured out how to change the threshold of classification in the original lda model object. So, I'll manually (get in there!) change the predictions to classify a win at a 60% threshold.

```
modified.preds <- predictions$posterior
modified.preds[modified.preds[,3] < 0.60] = 0
modified.preds[modified.preds[,3] >= 0.60] = 2
```

Now we compare this to the truth and nothing but the truth.

```
abs.diff <- abs(test.data$result - as.integer(modified.preds[,1]))
sum(abs.diff) / length(test.data$result)
```

```
## [1] 0.8309789
```

Ouch. We can think of this is as making a mistake 81 percent of the time. We could try to lower the threshold.

```
modified.preds <- predictions$posterior
modified.preds[modified.preds[,3] < 0.55] = 0
modified.preds[modified.preds[,3] >= 0.55] = 2
```

```
abs.diff <- abs(test.data$result - as.integer(modified.preds[,1]))
sum(abs.diff) / length(test.data$result)
```

```
## [1] 0.7295995
```

Muuuch better, but still pretty dismal.

```
modified.preds <- predictions$posterior
modified.preds[modified.preds[,3] < 0.52] = 0
modified.preds[modified.preds[,3] >= 0.52] = 2
```

```
abs.diff <- abs(test.data$result - as.integer(modified.preds[,1]))
sum(abs.diff) / length(test.data$result)
```

```
## [1] 0.6993518
```

Sweet. The only issue now is that we are almost right where we started. Since the original threshold of 0.5 was our genesis.

```
modified.preds <- predictions$posterior
modified.preds[modified.preds[,3] < 0.51] = 0
modified.preds[modified.preds[,3] >= 0.51] = 2
```

```
abs.diff <- abs(test.data$result - as.integer(modified.preds[,1]))
sum(abs.diff) / length(test.data$result)
```

```
## [1] 0.7030081
```

So, this is better than the 0.5 threshold, but still pretty bad. Maybe including another variable would make this model more awesome. Coming soon.