In previous blog posts we have discussed the [theory](http://thatdatatho.com/2018/02/13/linear-vs-quadratic-discriminant-analysis/) behind linear and quadratic discriminant analysis and we have also examined the [assumptions](http://thatdatatho.com/2018/02/27/assumption-checking-lda-vs-qda-r-tutorial-2/) for the Pima Indians [Data Set](https://www.kaggle.com/uciml/pima-indians-diabetes-database/data) with a [Shiny App.](http://thatdatatho.com/2018/04/22/86/)

In this blog post, we are going to implement these two algorithms and see which one performs better.

First, we are going to load all the required packages for this tutorial.

library(dplyr)

library(MASS)

library(ggplot2)

library(broom)

library(AUC)

diabetes <- read.csv("diabetes.csv")

head(diabetes, 3)

## Pregnancies Glucose BloodPressure SkinThickness Insulin BMI

## 1 6 148 72 35 0 33.6

## 2 1 85 66 29 0 26.6

## 3 8 183 64 0 0 23.3

dim(diabetes)

## [1] 768 9

diabetes <- dplyr::rename(diabetes, Pedigree = DiabetesPedigreeFunction)

diabetes[, 2:6][diabetes[, 2:6] == 0] <- NA # replaces all zero values from column two to six with NA

diabetes <- na.omit(diabetes) # now we omit all NA values

diabetes$Outcome <- ifelse(diabetes$Outcome == 1, "Diabetes", "No Diabetes")

After some data cleaning, where we have thrown out missing values, we are ready for our analysis.

**Linear Discriminant Analysis**

lda\_model <- lda(Outcome ~., data = diabetes)

preds <- predict(lda\_model)

head(preds$posterior)

## Diabetes No Diabetes

## 4 0.01976163 0.98023837

## 5 0.87150729 0.12849271

## 7 0.02576147 0.97423853

## 9 0.90400328 0.09599672

## 14 0.85368115 0.14631885

## 15 0.77561936 0.22438064

The output above shows the probabilities of being classified into the "Diabetes" or "No Diabetes" group. For example, observation one has not been tested positive for diabetes with a probability of 98%. Observation two has been diagnosed with diabetes with a probability of 87%. The model uses a 50% threshold for the posterior probabilities.

lda\_model

## Call:

## lda(Outcome ~ ., data = diabetes)

##

## Prior probabilities of groups:

## Diabetes No Diabetes

## 0.3316327 0.6683673

##

## Group means:

## Pregnancies Glucose BloodPressure SkinThickness Insulin

## Diabetes 4.469231 145.1923 74.07692 32.96154 206.8462

## No Diabetes 2.721374 111.4313 68.96947 27.25191 130.8550

## BMI Pedigree Age

## Diabetes 35.77769 0.6255846 35.93846

## No Diabetes 31.75076 0.4721679 28.34733

##

## Coefficients of linear discriminants:

## LD1

## Pregnancies -0.0576971998

## Glucose -0.0285466972

## BloodPressure -0.0002434335

## SkinThickness -0.0074724303

## Insulin 0.0005493946

## BMI -0.0415379420

## Pedigree -0.7002019020

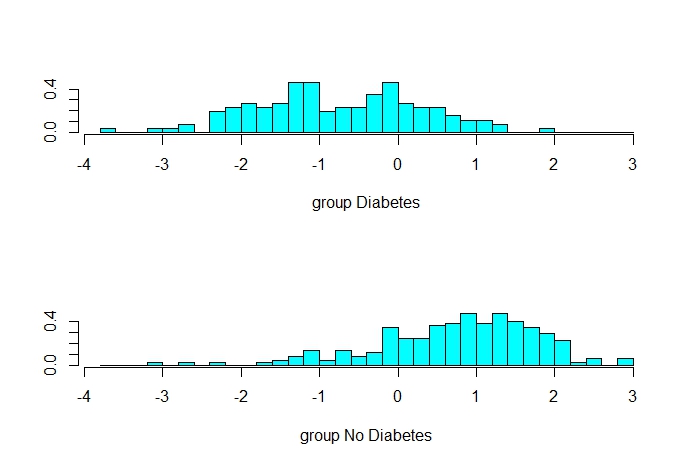
## Age -0.0261835760

From our output we can read off the prior probabilities $latex \hat{\Pi}\_1$ = 0.332 and $latex \hat{\Pi}\_2$ = 0.668. This means that around 33% of our data set includes people who have been diagnosed with diabetes and 66.8% who have not been diagnosed with diabetes.

Below the prior probabilities, we can see the group means of each predictor within each class. On average, people who have been diagnosed with diabetes were pregnant 4.5 times, have a glucose level of  145.2, a blood pressure of 74.1 (mm hg), triceps skinfold thickness of 33 mm, an insulin level of 207 mu U/ml, a BMI of 35.8 (weight in kg/(height in m)^2), a diabetes pedigree function of 0.63, and are 36 years old.

The scores below the group means are used to classify the observations into "Diabetes" and "No Diabetes".

plot(lda\_model)



More specifically, the scores, or coefficients of the linear discriminants output, are a linear combination that form the LDA decision rule. When the linear combination of these coefficients are negative, then the probability increases that an observation has diabetes (see plot), whereas when the linear combination is positive, an observation is more likely to belong to the "No Diabetes" group.

diabetes <- data.frame(diabetes, predicted = preds$class)

xtabs(~ predicted + Outcome, data = diabetes)

## Outcome

## predicted Diabetes No Diabetes

## Diabetes 76 30

## No Diabetes 54 232

# prediction accuracy

round((232+76)/(392), 4)

## [1] 0.7857

# error

round((30+54)/(392), 4)

## [1] 0.2143

We get a prediction error of about 21.4%. Our sensitivity (true positive rate) is only 76/(76+54) =  58.5%. We have a specificity (true negative rate) of  232/(232+30) = 88.5%.

**ROC Curve**

With the ROC curve, we can play a little bit with our sensitivity and specificity. Remember, we used a threshold of 50% for out posterior probabilities. The ROC curve is a plot of the true positive rate (TPR) versus false positive rate (FPR) as we vary the threshold. So we have a trade-off between true- and false-positive rates where we are able to change the threshold. The ideal threshold would be in the upper left corner of the ROC plot. Hence, we want to choose a threshold so that this particular point on the curve is closest to the upper left corner.

posterior\_yes <- preds$posterior[, "Diabetes"]

true\_yes <- (diabetes$Outcome == "Diabetes") %>%

as.numeric() %>%

factor()

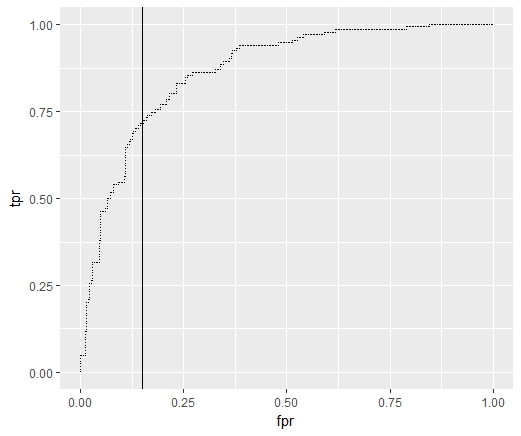
ROC\_res <- roc(posterior\_yes, true\_yes)

tidy\_ROC <- tidy(ROC\_res)

ggplot(tidy\_ROC, aes(x = fpr, y = tpr)) +

geom\_point(pch = ".") +

geom\_vline(xintercept = 0.15)



tidy\_ROC %>%

filter(fpr >= 0.148, fpr<=0.151)

## instance cutoff fpr tpr

## 1 283 0.3661268 0.148855 0.6923077

## 2 541 0.3637505 0.148855 0.7000000

## 3 244 0.3620957 0.148855 0.7076923

n <- nrow(diabetes)

thresh <- 0.3620957

dclass <- rep("No Diabetes", n)

dclass[posterior\_yes > thresh] <- "Diabetes"

outcome <- data.frame(diabetes,prednew = dclass)

xtabs(~ prednew + Outcome, data = outcome)

## Outcome

## prednew Diabetes No Diabetes

## Diabetes 92 39

## No Diabetes 38 223

(223+92)/(392)

## [1] 0.8035714

Above, we chose a threshold of around 0.36. Our true positive rate has increased from 76 to 92 observations that were correctly classified as the "Diabetes" group. Our false negative rate has decreased and we falsely classified 39 people into the "Diabetes" group, even though they have not been diagnosed with diabetes. Notice, that overall however, we have decreased our error rate.

**Quadratic Discriminant Analysis**

qda\_model <- qda(Outcome ~., data = diabetes)

preds <- predict(qda\_model)

head(preds$posterior)

## Diabetes No Diabetes

## 4 0.006150471 9.938495e-01

## 5 0.999980207 1.979306e-05

## 7 0.012858630 9.871414e-01

## 9 0.999993547 6.453313e-06

## 14 1.000000000 4.001310e-14

## 15 0.834364633 1.656354e-01

diabetes <- data.frame(diabetes, predicted = preds$class)

xtabs(~ predicted + Outcome, data = diabetes)

## Outcome

## predicted Diabetes No Diabetes

## Diabetes 86 32

## No Diabetes 44 230

(230+86)/(392)

## [1] 0.8061224

posterior\_yes <- preds$posterior[, "Diabetes"]

true\_yes <- (diabetes$Outcome == "Diabetes") %>%

as.numeric() %>%

factor()

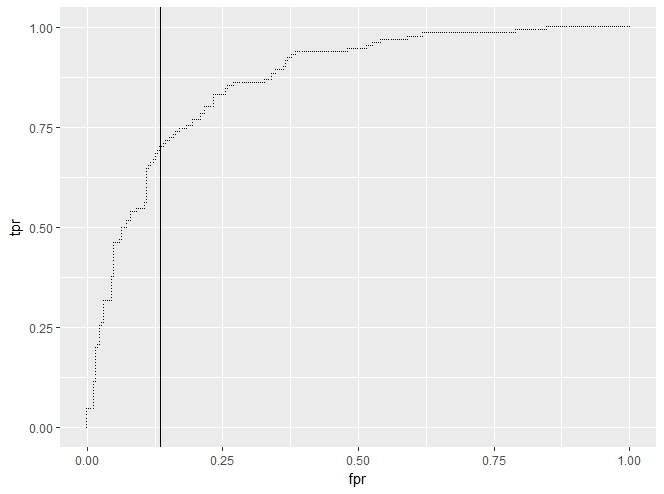
ROC\_res <- roc(posterior\_yes, true\_yes)

tidy\_ROC <- tidy(ROC\_res)

ggplot(tidy\_ROC, aes(x = fpr, y = tpr)) +

geom\_point(pch = ".") +

geom\_vline(xintercept = 0.135)



tidy\_ROC %>%

filter(fpr >= 0.13, fpr<=0.14)

## instance cutoff fpr tpr

## 1 748 0.4568396 0.1335878 0.6923077

## 2 32 0.4488076 0.1335878 0.7000000

## 3 391 0.4432782 0.1374046 0.7000000

n <- nrow(diabetes)

thresh <- 0.4432782

dclass <- rep("No Diabetes", n)

dclass[posterior\_yes > thresh] <- "Diabetes"

outcome <- data.frame(diabetes, prednew = dclass)

xtabs(~ prednew + Outcome, data = outcome)

## Outcome

## prednew Diabetes No Diabetes

## Diabetes 91 36

## No Diabetes 39 226

(91+226)/(392)

## [1] 0.8086735

For quadratic discriminant analysis, there is nothing much that is different from the linear discriminant analysis in terms of code. The quadratic discriminant analysis algorithm yileds the best classification rate. This might be due to the fact that the covariances matrices differ or because the true decision boundary is not linear.

We play again with the ROC curve and determine a threshold that yields us a better classification rate. Maybe you can achieve an even better one. Try it out!