Lab Assignment #7

Nick Noel & Liz Villa

Due March 24, 2023

Instructions

The purpose of this lab is to introduce several different classification strategies and variations on classification accuracy. In this lab we will work with another staple set of strategies: Naive Bayes and linear/quadratic discriminant analysis.

```
library(ISLR2)
library(ggplot2)
library(dplyr)
library(nycflights13)
library(e1071) # Naive Bayes
library(MASS) # LDA/QDA
library(yardstick) # only tidymodels package we'll need in this lab
```

This lab assignment is worth a total of **25 points**.

Problem 1: Naive Bayes

Part a (Code: 1 pt)

Run the code in ISLR Lab 4.7.5.

```
attach(Smarket)
train <- (Smarket$Year <2005)
Smarket.2005 <- Smarket[!train,]
Direction.2005 <- Direction[!train]
library(e1071)
```

```
nb.fit <- naiveBayes(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)


nb.fit
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##     Down       Up
## 0.491984 0.508016
##
```

```
## Conditional probabilities:
##       Lag1
## Y              [,1]     [,2]
##   Down  0.04279022 1.227446
##   Up   -0.03954635 1.231668
##
##       Lag2
## Y              [,1]     [,2]
##   Down  0.03389409 1.239191
##   Up   -0.03132544 1.220765
```

```
mean(Lag1[train][Direction[train] == "Down"])
```

```
## [1] 0.04279022
```

```
sd(Lag1[train][Direction[train] == "Down"])
```

```
## [1] 1.227446
```

```
nb.class <- predict(nb.fit,Smarket.2005)

table(nb.class, Direction.2005)
```

```
##          Direction.2005
## nb.class Down  Up
##     Down   28  20
##     Up     83 121
```

```
mean(nb.class == Direction.2005)
```

```
## [1] 0.5912698
```

```
nb.preds <- predict(nb.fit, Smarket.2005, type = "raw")
nb.preds[1:5,]
```

```
##           Down        Up
## [1,] 0.4873164 0.5126836
## [2,] 0.4762492 0.5237508
## [3,] 0.4653377 0.5346623
## [4,] 0.4748652 0.5251348
## [5,] 0.4901890 0.5098110
```

Part b (Code: 1 pt)

Recall that in Lab 6 we filtered the flights to only the United, American, and Delta carriers. Here we add another variable, `arr_ontime`, to represent whether the flight arrived on time or not.

```
flights2 <- flights %>% filter(
  carrier %in% c("UA", "AA", "DL"),
  !is.na(dep_delay),
  !is.na(arr_delay)
) %>%
  mutate(
    arr_ontime = as.factor(if_else(arr_delay <= 0, "yes", "no"))
  )
```

Non-randomly divide the `flights2` dataset into `flights_training`, which contains all flights through October, and `flights_test`, which contains all flights in November and December. You should be able to use the `filter` function to do this.

```r
flights2$arr_ontime <- relevel(flights2$arr_ontime, ref = "yes")
flights_training <- flights2 %>%
  filter(month == 10)
flights_test <- flights2 %>%
  filter(month == 11 | month == 12)
```

Then, fit a Naive Bayes model on the training set predicting whether a flight will be delayed (`arr_ontime` = "no") based on the departure delay (`dep_delay`), `carrier`, `distance` traveled, and `origin`.

```r
nb.flit <- naiveBayes(arr_ontime ~ dep_delay + carrier + distance +
                        origin, data = flights_training)
```

Part c (Code: 1.5 pts)

Unfortunately, there is no easy way to use `augment` on this model, so we'll have to make the predictions ourselves.

First, make class predictions on the `flights_test` dataset using similar code to that done in Lab 4.7.5. Then, create the `flights_nb_predictions` data frame or tibble containing two columns: `predicted`, representing the predicted classes, and `actual`, representing the actual classes. Use `flights_nb_predictions` to obtain the confusion matrix for the model.

```r
nb.flass <- predict(nb.flit, flights_test)

flights_nb_predictions <- data.frame(
  predicted = nb.flass,
  actual = flights_test$arr_ontime
)

(cmtrx <- conf_mat(flights_nb_predictions, truth = actual,
         estimate = predicted))
```

```
##             Truth
## Prediction    yes    no
##        yes 12787  5043
##        no    662  4206
```

Part d (Code: 0.5 pts; Explanation: 2 pts)

Without running any additional code, use the confusion matrix from part (c) to estimate the sensitivity, specificity, positive predictive value, and negative predictive value for the model. Express all answers as fractions and then convert to decimals rounded to the thousandths place (3 decimal places).

Sensitivity = 4206/9249 0.455

Specificity = 12787/13449 0.951

Positive Predictive Value = 4206/4868 0.864

Negative Predictive Value = 12787/17830 0.717

Then, using the `summary` function on your confusion matrix, check your answers. Remember that we are trying to predict that a flight will be delayed (`arr_ontime` = "no").

```r
summary(cmtrx, event_level = "second")
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
```

```
##  1 accuracy             binary         0.749
##  2 kap                  binary         0.438
##  3 sens                 binary         0.455
##  4 spec                 binary         0.951
##  5 ppv                  binary         0.864
##  6 npv                  binary         0.717
##  7 mcc                  binary         0.485
##  8 j_index              binary         0.406
##  9 bal_accuracy         binary         0.703
## 10 detection_prevalence binary         0.214
## 11 precision            binary         0.864
## 12 recall               binary         0.455
## 13 f_meas               binary         0.596
```

Looking at the summary, the sens row estimate matches our sensitivity, the spec row estimate matches our specificity, the ppv row estimate matches our positive predictive value, the npv row estimate matches our negative predictive value. :)

Problem 2: Discriminant Analysis

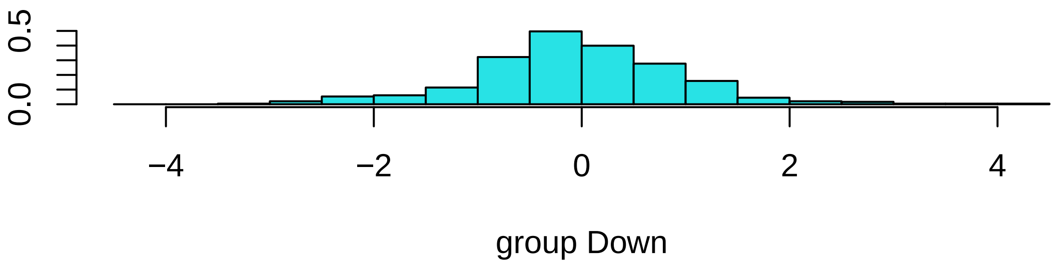
Part a (Code: 1 pt)

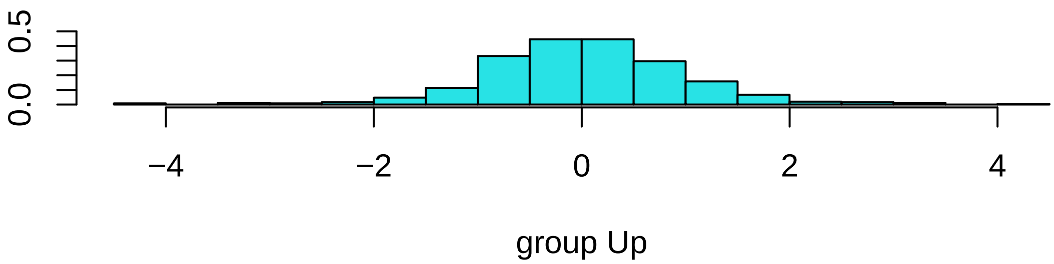
Run the code in ISLR Lab 4.7.3.

```r
lda.fit <- lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
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
```

```r
plot(lda.fit)
```






```
lda.pred <- predict(lda.fit, Smarket.2005)
names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```
lda.class <- lda.pred$class
table(lda.class, Direction.2005)
```

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

```
mean(lda.class == Direction.2005)
```

```
## [1] 0.5595238
```

```
sum(lda.pred$posterior[,1]>=.5)
```

```
## [1] 70
```

```
sum(lda.pred$posterior[,1]<.5)
```

```
## [1] 182
```

```
lda.pred$posterior[1:20,1]
```

```
##       999      1000      1001      1002      1003      1004      1005      1006
## 0.4901792 0.4792185 0.4668185 0.4740011 0.4927877 0.4938562 0.4951016 0.4872861
##      1007      1008      1009      1010      1011      1012      1013      1014
```

```
## 0.4907013 0.4844026 0.4906963 0.5119988 0.4895152 0.4706761 0.4744593 0.4799583
##      1015      1016      1017      1018
## 0.4935775 0.5030894 0.4978806 0.4886331
```

```
lda.class[1:20]
```

```
##  [1] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Down Up   Up   Up
## [16] Up   Up   Down Up   Up
## Levels: Down Up
```

```
sum(lda.pred$posterior[,1]>.9)
```

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

Part b (Code: 1 pt)

Run the code in ISLR Lab 4.7.4.

```
qda.fit <- qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

qda.fit
```

```
## Call:
## qda(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
```

```
qda.class <- predict(qda.fit, Smarket.2005)$class

table(qda.class, Direction.2005 )
```

```
##          Direction.2005
## qda.class Down  Up
##      Down   30  20
##      Up     81 121
```

```
mean(qda.class == Direction.2005)
```

```
## [1] 0.5992063
```

Part c (Code: 1 pt)

Fit a LDA model on the training set predicting whether a flight will be delayed (`arr_ontime` = “no”) based on the departure delay (`dep_delay`), `carrier`, `distance` traveled, and `origin`.

```
lda.flit.fit <-lda(arr_ontime ~ dep_delay + carrier + distance + origin, data = flights_training)

lda.flit.fit
```

```
## Call:
## lda(arr_ontime ~ dep_delay + carrier + distance + origin, data = flights_training)
##
```

```
## Prior probabilities of groups:
##       yes        no 
## 0.6946888 0.3053112 
## 
## Group means:
##     dep_delay carrierDL carrierUA distance originJFK originLGA
## yes -2.313854 0.3674215 0.4063794 1359.280 0.2949842 0.3308985
## no  20.626870 0.2969529 0.4700831 1428.542 0.2470914 0.3307479
## 
## Coefficients of linear discriminants:
##                     LD1
## dep_delay  0.0406933057
## carrierDL -0.2639537167
## carrierUA -0.0556703446
## distance   0.0002053784
## originJFK -0.2409482989
## originLGA  0.1864510004
```

Part d (Code: 1.5 pts)

Unfortunately, there is no easy way to use `augment` on this model, so we'll have to make the predictions ourselves.

First, make class predictions on the `flights_test` dataset using similar code to that done in Lab 4.7.3. Then, create the `flights_lda_predictions` data frame or tibble containing two columns: `predicted`, representing the predicted classes, and `actual`, representing the actual classes. Use `flights_lda_predictions` to obtain the confusion matrix for the model.

```
lda.flit.pred <- predict(lda.flit.fit, flights_test)


flights_lda_predictions <- data.frame(
  predicted = lda.flit.pred$class,
  actual = flights_test$arr_ontime
)

(cnfsnm <- conf_mat(flights_lda_predictions, truth = actual,
         estimate = predicted))
```

```
##           Truth
## Prediction   yes    no
##        yes 13377  6709
##        no     72  2540
```

Part e (Code: 0.5 pts; Explanation: 2 pts)

Without running any additional code, use the confusion matrix from part (d) to estimate the sensitivity, specificity, positive predictive value, and negative predictive value for the model. Express all answers as fractions and then convert to decimals rounded to the thousandths place (3 decimal places).

Sensitivity = 2541/9249 = 0.275

Specificity = 13377/13449 = 0.995

Positive Predictive Value = 2541/2613 = 0.972

Negative Predictive Value = 13377/20085 = 0.666

Then, using the `summary` function on your confusion matrix, check your answers. Remember that we are trying to predict that a flight will be delayed (`arr_ontime` = "no").

```
summary(cnfsnm, event_level = "second")
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.701
##  2 kap                  binary         0.303
##  3 sens                 binary         0.275
##  4 spec                 binary         0.995
##  5 ppv                  binary         0.972
##  6 npv                  binary         0.666
##  7 mcc                  binary         0.415
##  8 j_index              binary         0.269
##  9 bal_accuracy         binary         0.635
## 10 detection_prevalence binary         0.115
## 11 precision            binary         0.972
## 12 recall               binary         0.275
## 13 f_meas               binary         0.428
```

Looking at the summary, the sens row estimate matches our sensitivity, the spec row estimate matches our specificity, the ppv row estimate matches our positive predictive value, the npv row estimate matches our negative predictive value. :)

Part f (Code: 3 pts; Explanation: 2 pts)

Repeat parts (c) through (e) for the QDA model. (Obviously, call your new data frame/tibble `flights_qda_predictions` instead.)

(c)

```
qda.flit.fit <-qda(arr_ontime ~ dep_delay + carrier + distance + origin, data = flights_training)
```

(d)

```
qda.flit.pred <- predict(qda.flit.fit, flights_test)


flights_qda_predictions <- data.frame(
  predicted = qda.flit.pred$class,
  actual = flights_test$arr_ontime
)

(cnfsnmq <- conf_mat(flights_qda_predictions, truth = actual,
         estimate = predicted))
```

```
##           Truth
## Prediction   yes    no
##        yes 12801  4922
##        no    648  4327
```

Sensitivity = 4327/9249 = 0.468

Specificity = 12801/13449 = 0.952

Positive Predictive Value = 4327/4975 = 0.87

Negative Predictive Value = 12801/17723 = 0.722

```
summary(cnfsnmq, event_level = "second")
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.755
##  2 kap                  binary         0.452
##  3 sens                 binary         0.468
##  4 spec                 binary         0.952
##  5 ppv                  binary         0.870
##  6 npv                  binary         0.722
##  7 mcc                  binary         0.498
##  8 j_index              binary         0.420
##  9 bal_accuracy         binary         0.710
## 10 detection_prevalence binary         0.219
## 11 precision            binary         0.870
## 12 recall               binary         0.468
## 13 f_meas               binary         0.608
```

Looking at the summary, the sens row estimate matches our sensitivity, the spec row estimate matches our specificity, the ppv row estimate matches our positive predictive value, the npv row estimate matches our negative predictive value. :)

Problem 3: Model Selection

Part a (Code: 2 pts)

Add a column to the `flights_nb_predictions`, `flights_lda_predictions`, and `flights_qda_predictions` indicating the probability of not arriving on time (`arr_ontime == "no"`). It may be easiest to first obtain the predicted probabilities of being in each class, then add the column to the relevant data frame using `cbind` or `mutate`.

Then, compute the Brier scores for each model. As shown in the class activity, it is easiest to use the `mutate` function to obtain the "squared error" for each observation and then average the squared error.

```
nb.flassprob <- predict(nb.flit, flights_test, type = "raw")

flights_nb_predictions <- flights_nb_predictions %>%
  mutate(prob_not_aot = nb.flassprob[,"no"])


flights_lda_predictions <- flights_lda_predictions %>%
  mutate(prob_not_aot = lda.flit.pred$posterior[,2])

flights_qda_predictions <- flights_qda_predictions %>%
  mutate(prob_not_aot = qda.flit.pred$posterior[,2])
```

```
brier_nb <- flights_nb_predictions %>%
  mutate(squared_error = case_when(
    predicted == "yes" ~ (prob_not_aot)^2,
    predicted == "no" ~ (1 - prob_not_aot)^2))

brs_nb <- mean(brier_nb$squared_error)
```

```
brier_lda <- flights_lda_predictions %>%
  mutate(squared_error = case_when(
    predicted == "yes" ~ (prob_not_aot)^2,
    predicted == "no" ~ (1 - prob_not_aot)^2))

brs_lda <- mean(brier_lda$squared_error)

brier_qda <- flights_qda_predictions %>%
  mutate(squared_error = case_when(
    predicted == "yes" ~ (prob_not_aot)^2,
    predicted == "no" ~ (1 - prob_not_aot)^2))

brs_qda <- mean(brier_qda$squared_error)
```

Part b (Code: 1 pt)

Using the `mn_log_loss` function, obtain the cross-entropy/log loss for each of the three models.

```
lgl_nb <- mn_log_loss(flights_nb_predictions,
            truth = actual,
            prob_not_aot,
            event_level = "second")

lgl_lda <- mn_log_loss(flights_lda_predictions,
            truth = actual,
            prob_not_aot,
            event_level = "second")

lgl_qda <- mn_log_loss(flights_qda_predictions,
            truth = actual,
            prob_not_aot,
            event_level = "second")
```

Part c (Code: 1 pt)

Using the `mcc` function, obtain the Matthews Correlation Coefficient for each of the three models.

```
mcc_nb <- mcc(flights_nb_predictions, actual, predicted)

mcc_lda <- mcc(flights_lda_predictions, actual, predicted)

mcc_qda <- mcc(flights_qda_predictions, actual, predicted)
```

Part d (Explanation: 1.5 pts)

Compare the Brier score, log loss, and Matthews correlation coefficient for the three models by filling in the table below. Round all numbers to 3 decimal places.

Model	Brier	log loss	MCC
Naive Bayes	0.012	0.823	0.485
LDA	0.068	0.58	0.415
QDA	0.012	0.746	0.498

Which of the three models performs the best on this test set by each measure?

By Brier Scores, the Naive Bayes and QDA model are tied for best since we want lower Brier Scores.

By log loss, the LDA model performs best since we want a lower log loss.

By MCC, the QDA model performs best since we are looking for a higher correlation coefficient.

Part e (Explanation: 1.5 pts)

If you had to recommend one of the three models to use to predict whether a flight would be delayed, would you use the Naive Bayes, LDA, or QDA model? Explain.

If we can reasonably assume that our predictors are independent, I would use the Naive Bayes model since it is performing almost as well as the QDA model when measuring accuracy with Brier Scores and MCC and Naive Bayes is less computationally demanding. If we could not make this assumption, or we were working with small data, I would use the QDA model since it performs best according to both measures of accuracy, Brier Scores and MCC.