# Problem Set 7

## Classification Part 2

Alex Tomack

Due Date: 2023-03-31

# Getting Set Up

Open `RStudio` and create a new RMarkDown file ( `.Rmd` ) by going to `File -> New File -> R Markdown...` . Accept defaults and save this file as `[LAST NAME]_ps7.Rmd` to your `code` folder.

Copy and paste the contents of this file into your `[LAST NAME]_ps7.Rmd` file. Then change the `author: [YOUR NAME]` (line 4) to your name.

All of the following questions should be answered in this `.Rmd` file. There are code chunks with incomplete code that need to be filled in.

This problem set is worth 10 total points, plus three extra credit points. The point values for each question are indicated in brackets below. To receive full credit, you must both have the correct code **and include a comment describing what each line does**. In addition, some questions ask you to provide a written response in addition to the code. Furthermore, some of the code chunks are totally empty, requiring you to try writing the code from scratch. Make sure to comment each line, explaining what it is doing!

You are free to rely on whatever resources you need to complete this problem set, including lecture notes, lecture presentations, Google, your classmates…you name it. However, the final submission must be complete by you. There are no group assignments. To submit, compiled the completed problem set and upload the PDF file to Brightspace by midnight on 2023/03/31

**Good luck!**

# Question 0

Require `tidyverse` and `tidymodels` (for calculating AUC), and load the `admit_data.rds` (https://github.com/jbisbee1/DS1000_S2023/blob/main/Lectures/7_Classification/data/admit_data.rds?raw=true') data to an object called `ad` . (Tip: use the `read_rds()` function with the link to the raw data.)

```
require(tidyverse)
```

```
## Loading required package: tidyverse
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.4.0     ✓ purrr   1.0.0
## ✓ tibble  3.2.0     ✓ dplyr   1.1.0
## ✓ tidyr   1.2.1     ✓ stringr 1.5.0
## ✓ readr   2.1.3     ✓ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
require(tidymodels)
```

```
## Loading required package: tidymodels
## ── Attaching packages ──────────────────────────────────────── tidymodels 1.0.0 ──
## ✓ broom        1.0.2     ✓ rsample      1.1.1
## ✓ dials        1.1.0     ✓ tune         1.0.1
## ✓ infer        1.0.4     ✓ workflows    1.1.3
## ✓ modeldata    1.1.0     ✓ workflowsets 1.0.0
## ✓ parsnip      1.0.4     ✓ yardstick    1.1.0
## ✓ recipes      1.0.5
## ── Conflicts ─────────────────────────────────────── tidymodels_conflicts() ──
## ✖ scales::discard() masks purrr::discard()
## ✖ dplyr::filter()   masks stats::filter()
## ✖ recipes::fixed()  masks stringr::fixed()
## ✖ dplyr::lag()      masks stats::lag()
## ✖ yardstick::spec() masks readr::spec()
## ✖ recipes::step()   masks stats::step()
## • Search for functions across packages at https://www.tidymodels.org/find/
```

```
require(modelr)
```

```
## Loading required package: modelr
##
## Attaching package: 'modelr'
##
## The following objects are masked from 'package:yardstick':
##
##     mae, mape, rmse
##
## The following object is masked from 'package:broom':
##
##     bootstrap
```

```
ad<-read_rds('https://github.com/jbisbee1/DS1000_S2023/blob/main/Lectures/7_Classification/data/admit_data.rds?raw=true')
```

# Question 1 [3 points]

a. Compare a linear regression ( `mLM <- lm(...)` ) to a logit regression ( `mLG <- glm(...)` ) where you predict attendance ( `yield` ) as a function of the following $X$ predictors:

- `distance`
- `income`
- `sat`
- `gpa`
- `visit`
- `registered`
- `legacy`
- `net_price`

Evaluate the model performance using `roc_auc` based on cross validation with 100 iterations, using an 80-20% split of the data [2 points].

b. Does the linear regression model or the logit perform better? [1 point]

```
set.seed(123)
# a.
cvRes <- NULL
for(i in 1:100) {
  # Prepare the train and test datasets with sample()
  inds<-sample(1:nrow(ad), size=round(nrow(ad)*.8), replace=F)
  train<-ad%>%slice(inds)
  test<-ad%>%slice(-inds)
  # Linear
  mLM <- lm(formula=yield~distance+income+sat+gpa+visit+registered+legacy+net_price, data=train)
# Run linear regression on the train data

  # Logit
  mLG <- glm(formula=yield~distance+income+sat+gpa+visit+registered+legacy+net_price, data=trai
n, family=binomial(link="logit")) # Run logit regression on the train data

  toEval <- test %>%
    mutate(predLM = predict(mLM, newdata=test), # Predict the linear regression output
           predLG = predict(mLG, newdata=test, type="response")) %>% # Predict the logit regress
ion output
    mutate(yield=factor(yield, c("1", "0"))) # Prepare the yield outcome to have the '1' occur f
irst

  tmpLM <- roc_auc(data=toEval, truth="yield", estimate="predLM") %>% # Calculate AUC for the li
near regression
    mutate(cvInd = i,
           algo = 'LM')

  tmpLG <- roc_auc(data=toEval, truth="yield", estimate="predLG") %>% # Calculate AUC for the lo
git regression
    mutate(cvInd = i,
           algo = 'Logit')


  # Append to the cvRes object to save
  cvRes<-tmpLM%>%
    bind_rows(tmpLG)%>%
    bind_rows(cvRes)
}

# b. Calculate average AUC by regression type
cvRes%>%
  group_by(algo)%>%
  mutate(algoAvg=mean(.estimate))
```

```
## # A tibble: 200 × 6
## # Groups:   algo [2]
##    .metric .estimator .estimate cvInd algo  algoAvg
##    <chr>   <chr>          <dbl> <int> <chr>   <dbl>
##  1 roc_auc binary         0.904   100 LM      0.872
##  2 roc_auc binary         0.930   100 Logit   0.909
##  3 roc_auc binary         0.866    99 LM      0.872
##  4 roc_auc binary         0.910    99 Logit   0.909
##  5 roc_auc binary         0.871    98 LM      0.872
##  6 roc_auc binary         0.908    98 Logit   0.909
##  7 roc_auc binary         0.886    97 LM      0.872
##  8 roc_auc binary         0.909    97 Logit   0.909
##  9 roc_auc binary         0.885    96 LM      0.872
## 10 roc_auc binary         0.918    96 Logit   0.909
## # … with 190 more rows
```

- b. Logistic regression tends to do better in our 100-fold CV– predicting ~3% more of the school's yield than a linear model, and predicting 90% of yield on average.

# Question 2 [3 points]

a. Based on the result to question 1, choose the best classification algorithm and train it on the full data. [1 point]

b. Calculate the specificity and sensitivity across different thresholds ranging from zero to one, and plot these as different colored lines. [1 point]

c. What is the optimal threshold to balance the trade-off between sensitivity and specificity based on this plot? **HINT**: Use `geom_vline()` and test different `xintercept` values until you nail the intersection between the two lines. [1 point]

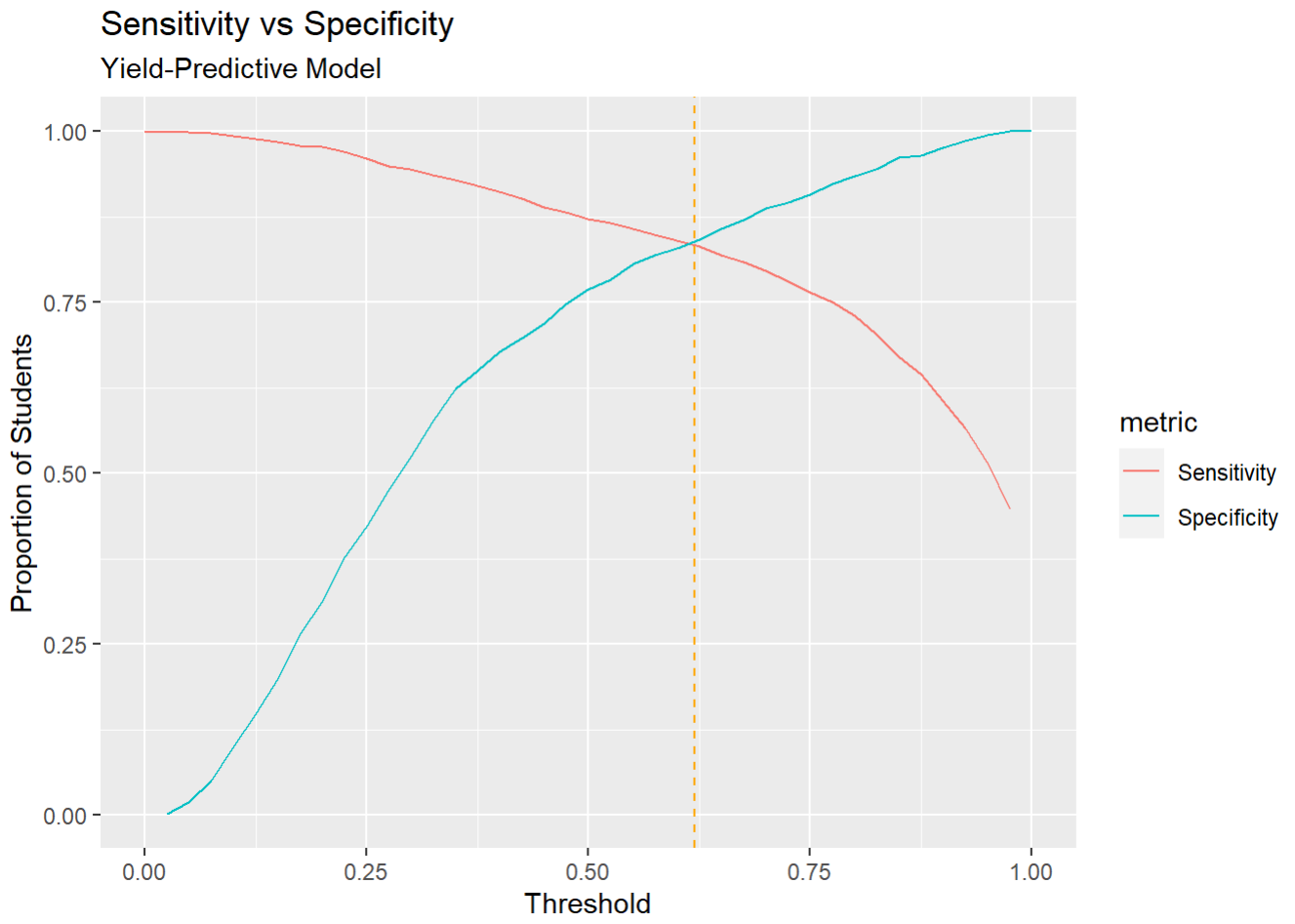
```
# a. Re-run the best regression model on the full data
mLG2<- glm(formula=yield~sat+legacy+visit+registered+income+gpa+distance+net_price, data=ad, fam
ily=binomial(link="logit"))

# b. Calculate sensitivity and specificity
ad <- ad %>%
  mutate(preds = predict(mLG2, type="response")) # Calculate predicted values

# Look over the threshold values between 0 and 1 by 0.025 and calculate proportion of correctly
predicted students
toplot <- NULL
for(thresh in seq(0,1, by=.025)) {
  tmp <- ad %>%
  mutate(pred_attend = ifelse(preds>thresh, 1, 0)) %>% # Calculate predicted attendance if the p
robability is greater than the threshold
  group_by(yield) %>%
  mutate(total_attend=n()) %>% # Calculate total students by whether they attended or not
  group_by(yield, pred_attend, total_attend) %>%
  summarise(nStudents=n(),.groups="drop") %>% # Calculate number of students by whether they did
/ did not attend, and whether they were predicted to attend
  mutate(prop = nStudents/total_attend) %>% # Calculate proportion of students that fall into ea
ch bin
  ungroup() %>%
  mutate(threshold = thresh) # Save the indicator for the threshold value

  toplot<-toplot%>%
    bind_rows(tmp) # Add it to the toplot object
}

toplot %>%
  mutate(metric = ifelse(yield==0&pred_attend==0, "Specificity",
                         ifelse(yield==1&pred_attend==1, "Sensitivity",NA))) %>%
  drop_na(metric) %>%
  ggplot(aes(x=threshold, y=prop, color=metric)) + # Plot the proportion versus the metric (use
color = metric)
  geom_line() +
  geom_vline(xintercept=0.62, linetype="dashed", color="orange") + # c. Try different values of
the xintercept until you find where the lines intersect
  labs(title = 'Sensitivity vs Specificity', # Make sure to label clearly!
       subtitle = 'Yield-Predictive Model',
       x = 'Threshold',
       y = 'Proportion of Students')
```




- c. The optimal tradeoff between sensitivity and specificity exists at a threshold of 0.62.

# Question 3 [4 points]

a. How many students with SAT scores higher than 1300 are currently enrolled ( `yield` )? How many students with SAT scores higher than 1300 are predicted to enroll according to our model? [1 point]

b. What is the average SAT score and total tuition among enrolled students? [1 point]

c. Reduce the net price ( `net_price` ) for students with SAT scores higher than 1300 by $5,000. How many are now estimated to enroll? [1 point]

d. What is the average SAT score among students predicted to enroll after adjusting the `net_price` ? What is the total tuition? [1 point]

```
# a. Look at students whose yield==1 and sat>=1300
ad%>%
  mutate(highsat=ifelse(yield==1&sat>=1300, 1, 0))%>%
  group_by(yield, highsat)%>%
  summarise(total_highsat=sum(highsat), .groups="drop")
```

```
## # A tibble: 3 × 3
##   yield highsat total_highsat
##   <int>   <dbl>         <dbl>
## 1     0       0             0
## 2     1       0             0
## 3     1       1           314
```

```
# create hypothetical data with 1300+ students
hypo_data<-ad %>%
  mutate(pred_sat=ifelse(sat>=1300, 1,0))

hypo_data<-hypo_data%>%
  mutate(prob_attend = predict(mLG, newdata=hypo_data, type="response"))%>%
  mutate(pred_attend=ifelse(prob_attend>.5, 1, 0))

hypo_data%>%
  filter(pred_attend==1)%>%
  summarise(tot_highsat=sum(pred_sat),
            tot_student=n())
```

```
##   tot_highsat tot_student
## 1         335        1447
```

```
# b. what is the average sat score and total tuition among enrolled students?
ad%>%
  filter(yield==1)%>%
  summarise(total_attend=n(),
            mean_sat=mean(sat),
            total_tuition=sum(tuition))
```

```
##   total_attend mean_sat total_tuition
## 1         1466 1225.941      65970000
```

```
# c. Reduce the net price (`net_price`) for students with SAT scores higher than 1300 by $5,000.
How many are now estimated to enroll?

# estimate with model using data_grid
red_data<-ad %>%
  mutate(net_price=ifelse(sat>=1300, net_price-5000, net_price))

red_data<-red_data%>%
  mutate(prob_attend = predict(mLG, newdata=red_data, type="response"))%>%
  mutate(pred_attend=ifelse(prob_attend>.5, 1, 0))

# evaluate whether we achieved our goal
red_data%>%
  filter(pred_attend==1)%>%
  summarise(sat_avg=mean(sat, na.rm=T),
            tot_student=n())
```

```
##   sat_avg tot_student
## 1 1234.24        1447
```

- a. 314 students with 1300+ SAT scores are currently enrolled, and 335 are predicted to enroll.
- b. of the 1466 students, the average SAT score is 1225 and the total tuition is $65,970,000.
- c. Implementing the reduced price, we see that 1447 students are predicted to enroll– boosting our yield rate with the predicted addition of 1000 students.
- d. The average SAT is boosted, too, sitting at 1234– a 10pt increase to the previous mean SAT.

# Extra Credit= [3 points]

a. How high can you increase the average SAT score while maintaining current revenues, using only the `net_price` to induce changes? [1 point]

b. Answer this question using a loop. [1 point]

c. How does your answer change if you restrict the final `net_price` value per observation to be no lower than zero, and no higher than $45,000? [1 point]

```
# INSERT CODE HERE
```

a. Write answer here.

b. Write answer here.