Problem Set 7

Classification Part 2

[YOUR NAME]

Due Date: 2023-11-10

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. Unlike the first two problem sets, 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 2023/11/10 Also note that the TAs and professors will not respond to Campuswire posts after 5PM on Friday, so don't wait until the last minute to get started!

Good luck!

ChatGPT Link [Optional]

*Copy the link to ChatGPT you used here: _______________________.

Question 0

Require `tidyverse` and `tidymodels` (for calculating AUC), and load the `admit_data.rds` (https://github.com/jbisbee1/DS1000_F2023/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 3.3.6      ✓ purrr   0.3.4
## ✓ tibble  3.1.8      ✓ dplyr   1.0.10
## ✓ tidyr   1.2.1      ✓ stringr 1.4.1
## ✓ readr   2.1.2      ✓ 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        1.0.1     ✓ rsample      1.1.0
## ✓ dials        1.0.0     ✓ tune         1.0.0
## ✓ infer        1.0.3     ✓ workflows    1.1.0
## ✓ modeldata    1.0.1     ✓ workflowsets 1.0.0
## ✓ parsnip      1.0.2     ✓ yardstick    1.1.0
## ✓ recipes      1.0.1
## ── 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()
## • Use tidymodels_prefer() to resolve common conflicts.
```

```
ad <- read_rds('https://github.com/jbisbee1/DS1000_F2023/blob/main/Lectures/7_Classification/dat
a/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) {
  inds <- sample(1:nrow(ad),size = round(nrow(ad)*.8),replace = F)
  train <- ad %>% slice(inds)
  test <- ad %>% slice(-inds)

  # Linear
  mLM <- lm(yield ~ distance + income + sat + gpa + visit + registered + legacy + net_price,trai
n)

  # Logit
  mLG <- glm(yield ~ distance + income + sat + gpa + visit + registered + legacy + net_price,fam
ily = binomial(link = 'logit'),data = train)

  toEval <- test %>%
    mutate(predLM = predict(mLM,newdata = test),
           predLG = predict(mLG,newdata = test,type = 'response')) %>%
    mutate(truth = factor(yield,levels = c('1','0')))

  tmpLM <- roc_auc(toEval,truth = 'truth',estimate = 'predLM') %>%
    mutate(cvInd = i,
           algo = 'LM')

  tmpLG <- roc_auc(toEval,truth = 'truth',estimate = 'predLG') %>%
    mutate(cvInd = i,
           algo = 'Logit')

  cvRes <- cvRes %>%
    bind_rows(tmpLM) %>%
    bind_rows(tmpLG)
}

# b.
cvRes %>%
  group_by(algo) %>%
  summarise(auc = mean(.estimate))
```

```
## # A tibble: 2 × 2
##   algo    auc
##   <chr> <dbl>
## 1 LM    0.872
## 2 Logit 0.909
```

- Based on this analysis, the logistic regression model performs better, with an AUC of 0.91 versus 0.87 for the linear model.

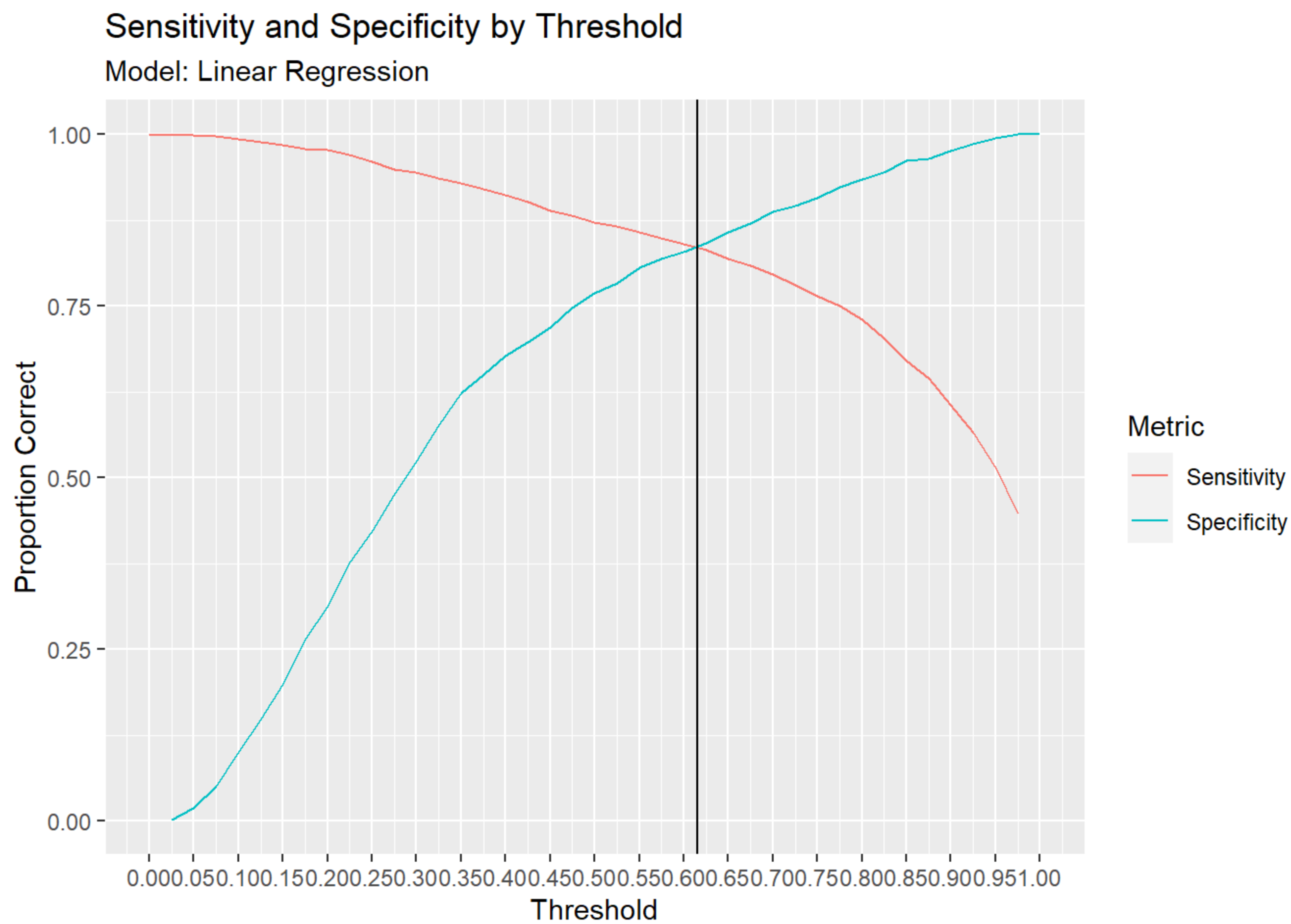
Question 2 [3 points]

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.
mFinal <- glm(yield ~ distance + income + sat + gpa + visit + registered + legacy + net_price,a
d,
                   family = binomial(link = 'logit'))

# b.
ad <- ad %>%
  mutate(preds = predict(mFinal,type = 'response'))

toplot <- NULL
for(thresh in seq(0,1,by = 0.025)) {
  toplot <- ad %>%
  mutate(pred_attend = ifelse(preds > thresh,1,0)) %>%
  group_by(yield) %>%
  mutate(total_attend = n()) %>%
  group_by(yield,pred_attend,total_attend) %>%
  summarise(nStudents=n(),.groups = 'drop') %>%
  mutate(prop = nStudents / total_attend) %>%
  ungroup() %>%
  mutate(accuracy = sum((yield == pred_attend)*nStudents) / sum(nStudents)) %>%
  mutate(threshold = thresh) %>%
    bind_rows(toplot)
}

toplot %>%
  mutate(metric = ifelse(yield == 1 & pred_attend == 1,'Sensitivity',
                         ifelse(yield == 0 & pred_attend == 0,'Specificity',NA))) %>%
  drop_na(metric) %>%
  ggplot(aes(x = threshold,y = prop,color = metric)) +
  geom_line() +
  scale_x_continuous(breaks = seq(0,1,by = .05)) +
  geom_vline(xintercept = .615) + # c.
  labs(title = 'Sensitivity and Specificity by Threshold',
       subtitle = 'Model: Linear Regression',
       x = 'Threshold',
       y = 'Proportion Correct',
       color = 'Metric')
```



- The optimal threshold is 0.615. This threshold value maximizes both sensitivity and specificity of the model.

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. 314 high SAT students currently enrolled.
ad %>%
  count(yield,sat > 1300)
```

```
##   yield sat > 1300    n
## 1     0      FALSE  655
## 2     0       TRUE   29
## 3     1      FALSE 1152
## 4     1       TRUE  314
```

```
# 328 high SAT students predicted to enroll.
ad %>%
  mutate(preds = predict(mFinal,type = 'response')) %>%
  mutate(pred_attend = ifelse(preds > .615,1,0)) %>%
  count(pred_attend,sat > 1300)
```

```
##   pred_attend sat > 1300    n
## 1           0      FALSE  799
## 2           0       TRUE   15
## 3           1      FALSE 1008
## 4           1       TRUE  328
```

```
# b.
ad %>%
  filter(yield == 1) %>%
  summarise(avgSAT = mean(sat,na.rm=T),
            tuition = sum(net_price,na.rm=T))
```

```
##     avgSAT  tuition
## 1 1225.941 30674149
```

```
# c.
hypo <- ad %>%
  mutate(net_price = ifelse(sat > 1300,net_price - 5000,net_price))

hypo %>%
  mutate(preds = predict(mFinal,newdata = hypo,type = 'response')) %>%
  mutate(pred_attend = ifelse(preds > .615,1,0)) %>%
  count(pred_attend,sat > 1300)
```

```
##   pred_attend sat > 1300    n
## 1           0      FALSE  799
## 2           0       TRUE   11
## 3           1      FALSE 1008
## 4           1       TRUE  332
```

```
# d.
hypo %>%
  mutate(preds = predict(mFinal,newdata = hypo,type = 'response')) %>%
  mutate(pred_attend = ifelse(preds > .615,1,0)) %>%
  filter(pred_attend == 1) %>%
  summarise(avgSAT = mean(sat,na.rm=T),
            tuition = sum(net_price,na.rm=T))
```

```
##     avgSAT  tuition
## 1 1238.878 25435144
```

a. There are 314 students currently enrolled with SAT scores higher than 1300. Our model predicts that there should be 328 students enrolled who have SAT scores higher than 1300.
b. The average SAT score is roughly 1226, and the total tuition is $30.7m.
c. After adjusting the price, the model predicts that 332 students with SAT scores greater than 1300 will enroll.
d. After adjusting the price, the model predicts that the average SAT score will be 1239 and the total tuition will be $25.4m.

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]

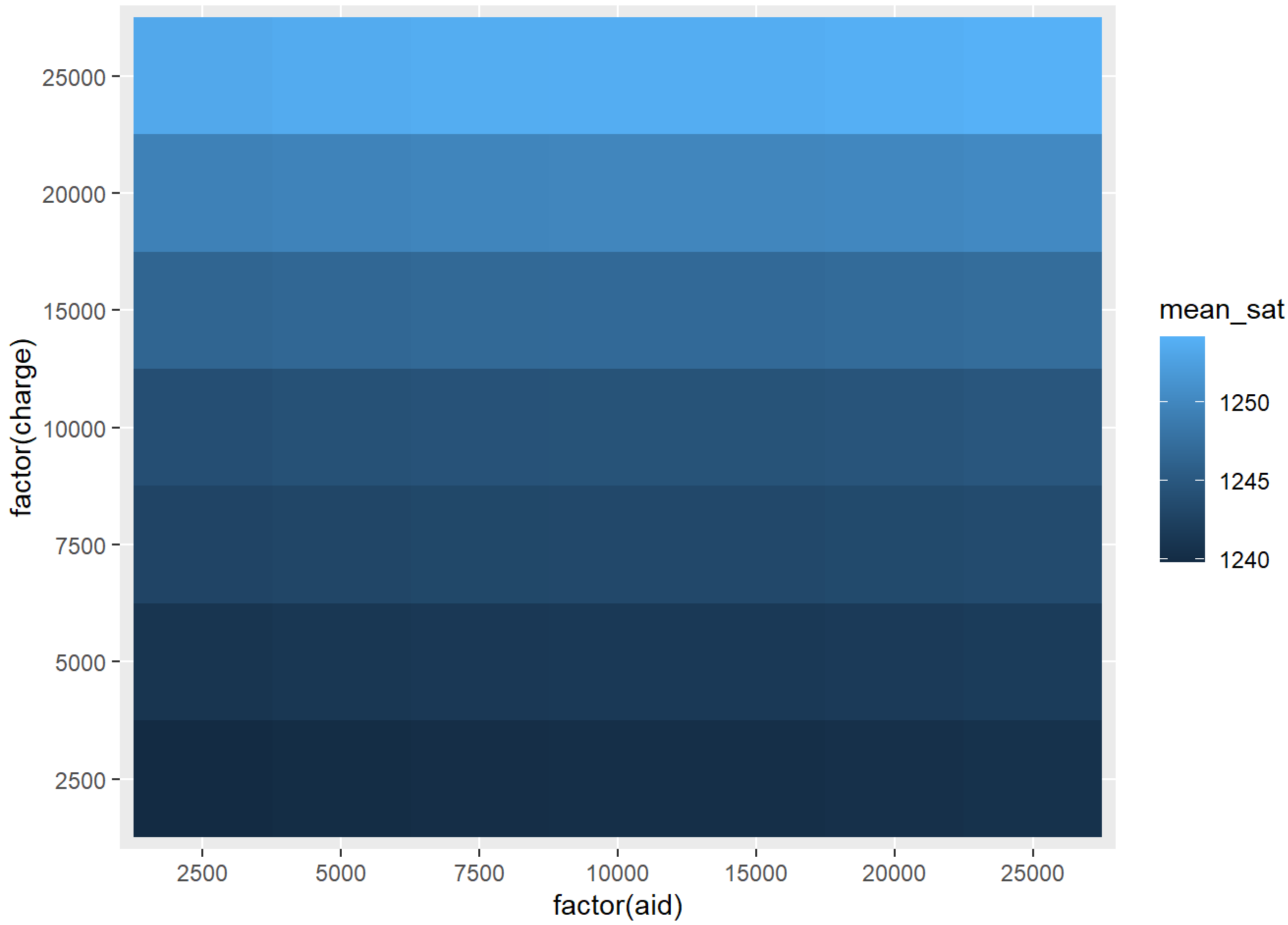
```
# a.
hypo <- ad %>%
  mutate(net_price = ifelse(sat >= 1300,0,45000))

hypo %>%
  mutate(preds = predict(mFinal,newdata = hypo,type = 'response')) %>%
  mutate(pred_attend = ifelse(preds > .615,1,0)) %>%
  filter(pred_attend == 1) %>%
  summarise(mean_sat = mean(sat),
            revenues = sum(net_price),
            nStudents = n())
```

```
##   mean_sat revenues nStudents
## 1 1238.299 35505000      1117
```

```r
# b.
toplot <- NULL
for(i in c(2500,5000,7500,10000,15000,20000,25000)) {
  for(j in c(2500,5000,7500,10000,15000,20000,25000)) {
    for(realistic in c(TRUE,FALSE)) { # c.
        hypo <- ad %>%
          mutate(net_price = ifelse(sat >= 1300,net_price - i,net_price + j))

        # c.
        if(realistic) {
                hypo <- hypo %>%
                   mutate(net_price = ifelse(net_price < 0,0,
                                    ifelse(net_price > 45000,45000,net_price)))
        }

        toplot <- hypo %>%
          mutate(preds = predict(mFinal,newdata = hypo,type = 'response')) %>%
          mutate(pred_attend = ifelse(preds > .615,1,0)) %>%
          filter(pred_attend == 1) %>%
          summarise(mean_sat = mean(sat),
                    revenues = sum(net_price),
                    nStudents = n()) %>%
          ungroup() %>%
          mutate(aid = i,
                 charge = j,
                 realistic = realistic) %>%
          bind_rows(toplot)

    }
  }
}

toplot %>%
  filter(charge == 25000)
```

```
##    mean_sat revenues nStudents   aid charge realistic
## 1  1254.217 29564701      1054 25000  25000     FALSE
## 2  1238.955 34619300      1172 25000  25000      TRUE
## 3  1254.021 31304701      1052 20000  25000     FALSE
## 4  1238.955 34619300      1172 20000  25000      TRUE
## 5  1253.841 33039701      1049 15000  25000     FALSE
## 6  1238.955 34619300      1172 15000  25000      TRUE
## 7  1253.841 34714701      1049 10000  25000     FALSE
## 8  1238.955 34825401      1172 10000  25000      TRUE
## 9  1253.792 35559701      1048  7500  25000     FALSE
## 10 1238.955 35387140      1172  7500  25000      TRUE
## 11 1253.605 36404701      1046  5000  25000     FALSE
## 12 1238.955 36025537      1172  5000  25000      TRUE
## 13 1253.358 37242201      1043  2500  25000     FALSE
## 14 1238.955 36696639      1172  2500  25000      TRUE
```

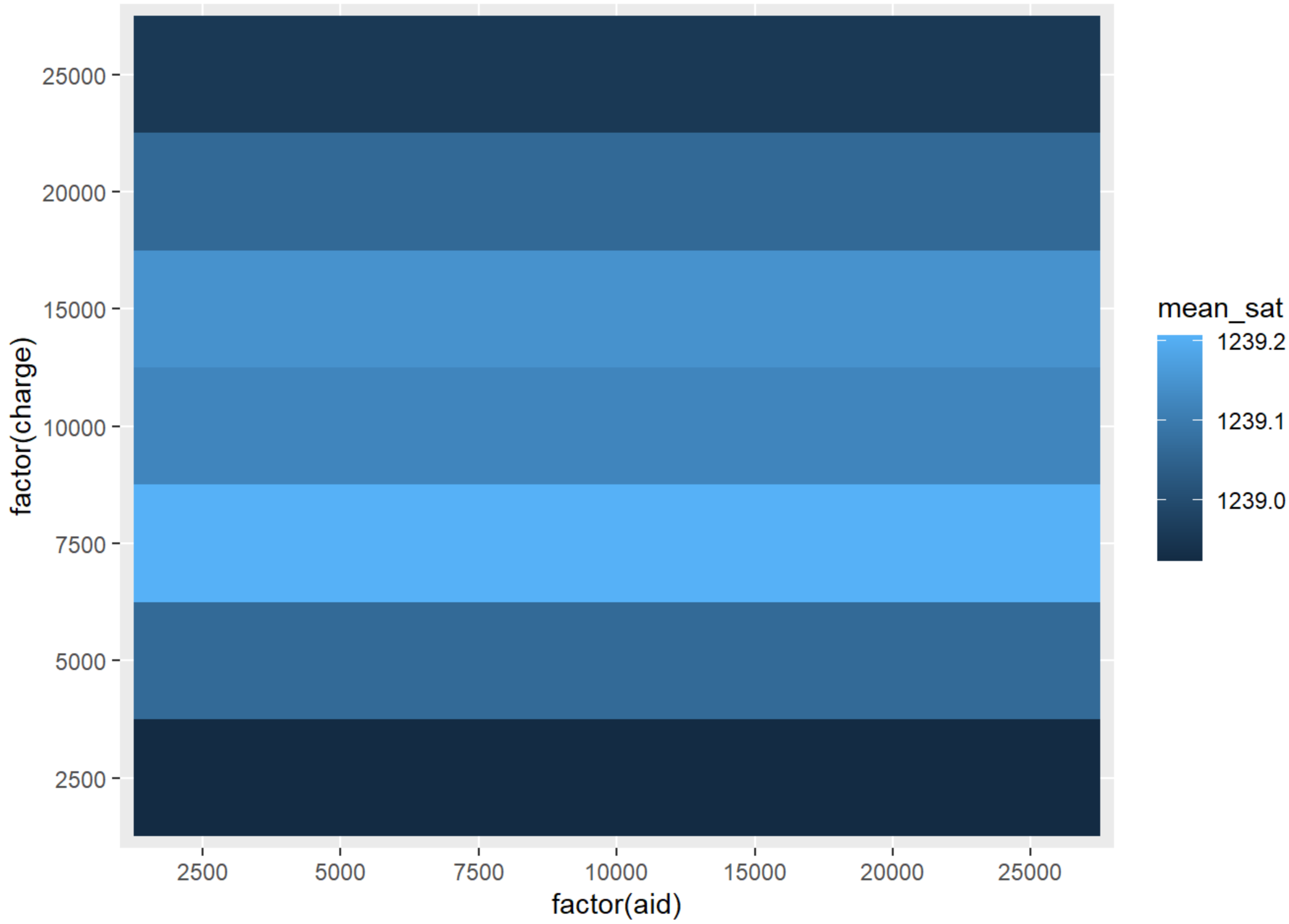
```
# b.
toplot %>%
  filter(!realistic) %>%
  ggplot(aes(x = factor(aid),y = factor(charge),fill = mean_sat)) +
  geom_tile()
```




```
toplot %>%
  filter(!realistic) %>%
  filter(revenues >= 30674149) %>%
  arrange(desc(mean_sat))
```

```
##    mean_sat revenues nStudents   aid charge realistic
## 1  1254.021 31304701      1052 20000  25000     FALSE
## 2  1253.841 33039701      1049 15000  25000     FALSE
## 3  1253.841 34714701      1049 10000  25000     FALSE
## 4  1253.792 35559701      1048  7500  25000     FALSE
## 5  1253.605 36404701      1046  5000  25000     FALSE
## 6  1253.358 37242201      1043  2500  25000     FALSE
## 7  1249.891 32292349      1116 15000  20000     FALSE
## 8  1249.891 33967349      1116 10000  20000     FALSE
## 9  1249.842 34812349      1115  7500  20000     FALSE
## 10 1249.659 35657349      1113  5000  20000     FALSE
## 11 1249.416 36494849      1110  2500  20000     FALSE
## 12 1246.920 32184522      1175 10000  15000     FALSE
## 13 1246.871 33029522      1174  7500  15000     FALSE
## 14 1246.692 33874522      1172  5000  15000     FALSE
## 15 1246.454 34712022      1169  2500  15000     FALSE
## 16 1244.211 31416657      1228  5000  10000     FALSE
## 17 1243.977 32254157      1225  2500  10000     FALSE
## 18 1242.796 30985783      1253  2500   7500     FALSE
```

```
# c.
toplot %>%
  filter(realistic) %>%
  ggplot(aes(x = factor(aid),y = factor(charge),fill = mean_sat)) +
  geom_tile()
```




```
toplot %>%
  filter(realistic) %>%
  filter(revenues >= 30674149) %>%
  arrange(desc(mean_sat))
```

```
##    mean_sat revenues nStudents   aid charge realistic
## 1  1239.146 31579121      1246 25000  15000      TRUE
## 2  1239.146 31579121      1246 20000  15000      TRUE
## 3  1239.146 31579121      1246 15000  15000      TRUE
## 4  1239.146 31785223      1246 10000  15000      TRUE
## 5  1239.146 32346961      1246  7500  15000      TRUE
## 6  1239.146 32985359      1246  5000  15000      TRUE
## 7  1239.146 33656460      1246  2500  15000      TRUE
## 8  1239.121 30834685      1279  5000  10000      TRUE
## 9  1239.121 31505787      1279  2500  10000      TRUE
## 10 1239.061 33356948      1211 25000  20000      TRUE
## 11 1239.061 33356948      1211 20000  20000      TRUE
## 12 1239.061 33356948      1211 15000  20000      TRUE
## 13 1239.061 33563050      1211 10000  20000      TRUE
## 14 1239.061 34124788      1211  7500  20000      TRUE
## 15 1239.061 34763185      1211  5000  20000      TRUE
## 16 1239.061 35434287      1211  2500  20000      TRUE
## 17 1238.955 34619300      1172 25000  25000      TRUE
## 18 1238.955 34619300      1172 20000  25000      TRUE
## 19 1238.955 34619300      1172 15000  25000      TRUE
## 20 1238.955 34825401      1172 10000  25000      TRUE
## 21 1238.955 35387140      1172  7500  25000      TRUE
## 22 1238.955 36025537      1172  5000  25000      TRUE
## 23 1238.955 36696639      1172  2500  25000      TRUE
```

a. I can achieve an average SAT score of 1254 while maintaining revenues above $3.07m if I reduce `net_price` by $20,000 for students with SAT scores greater than or equal to 1300, and charging those with SAT scores lower than 1300 an additional $25,000.

b. If I restrict `net_price` to be realistic values (i.e., never going lower than zero and never greater than the maximum tuition of $45,000), I can only improve average SAT scores to 1239 which is achieved by charging students with scores less than 1300 an additional $15,000 and incentivizing those with scores 1300 or greater with any amount of money.