Lecture notes

Prof. Bisbee, Vanderbilt University

2023-03-20

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.3.6     ✓ purrr   0.3.4
## ✓ tibble  3.1.7     ✓ dplyr   1.0.9
## ✓ tidyr   1.2.0     ✓ stringr 1.4.0
## ✓ readr   2.1.2     ✓ forcats 0.5.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
```

```
library(scales)
```

```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

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

```
## Rows: 2,150
## Columns: 14
## $ ID          <chr> "0001", "0002", "0003", "0004", "0005", "0006", "0007", "0…
## $ income      <dbl> 289720.59, 176763.29, 81204.02, 93320.52, 144991.22, 72720…
## $ sat         <dbl> 1107.403, 1387.607, 1000.000, 1134.883, 1202.686, 1053.033…
## $ gpa         <dbl> 3.597153, 4.000000, 3.072323, 3.682776, 3.970005, 3.474787…
## $ visit       <dbl> 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0…
## $ legacy      <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1…
## $ registered  <dbl> 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1…
## $ sent_scores <dbl> 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ distance    <dbl> 10.23279, 89.75984, 152.29961, 317.50274, 240.30712, 63.07…
## $ tuition     <dbl> 45000, 45000, 45000, 45000, 45000, 45000, 45000, 45000, 45…
## $ need_aid    <dbl> 0.000, 0.000, 8488.293, 3338.779, 0.000, 12093.802, 15156.…
## $ merit_aid   <dbl> 0.00, 35190.18, 0.00, 0.00, 30567.16, 0.00, 31633.66, 3219…
## $ net_price   <dbl> 45000.000, 9809.815, 36511.707, 41661.221, 14432.840, 3290…
## $ yield       <int> 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0…
```

```
ad %>%
  summarise(YieldRate = mean(yield))
```

```
##   YieldRate
## 1 0.6818605
```

```
ad %>%
  group_by(legacy) %>%
  summarise(prob_attend = mean(yield))
```

```
## # A tibble: 2 × 2
##   legacy prob_attend
##    <dbl>       <dbl>
## 1      0       0.641
## 2      1       0.780
```

```
# do this for visit & sent_scores
```

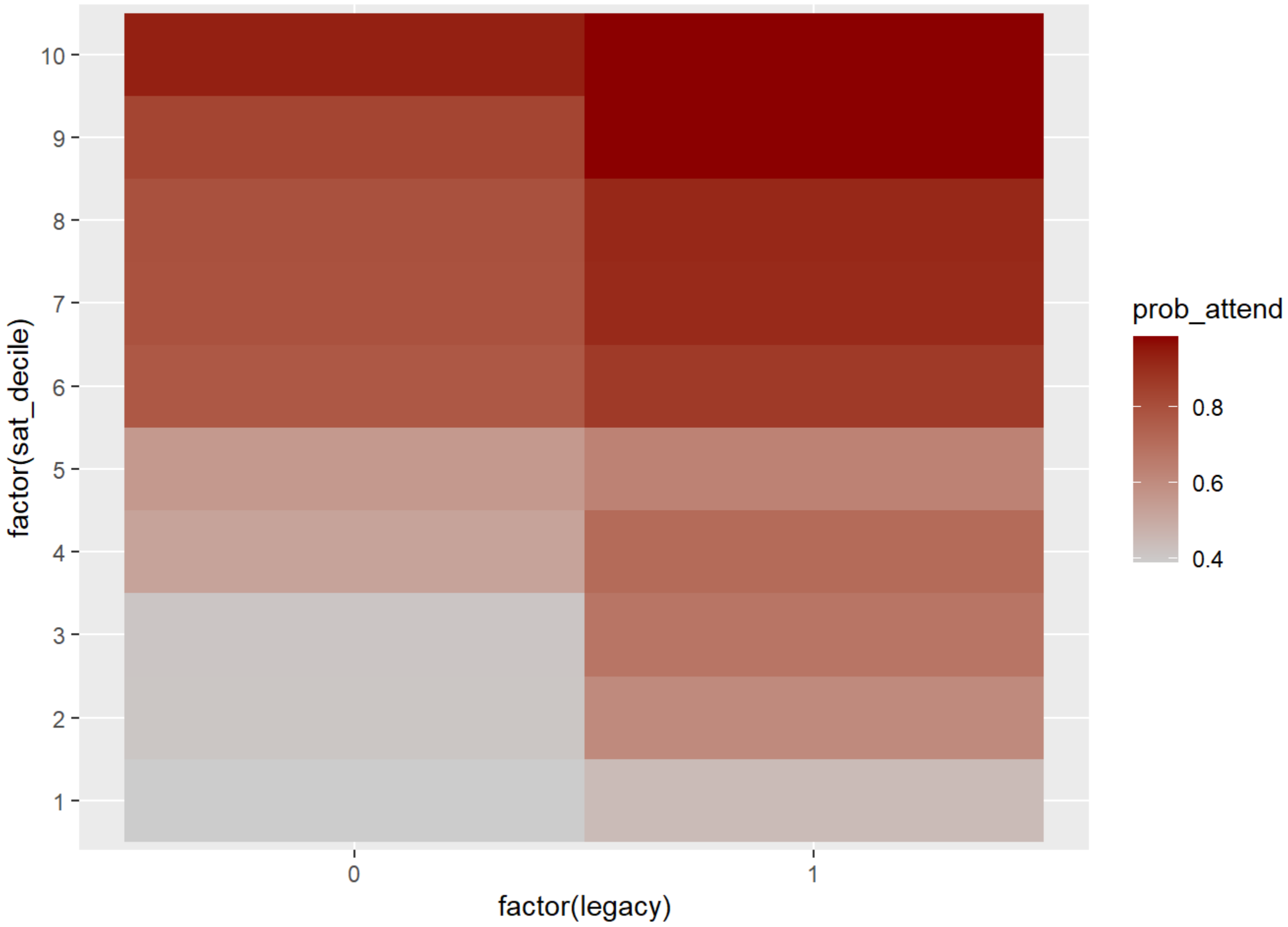
Heatmap

- `ntile()` function

```
toplot <- ad %>%
  mutate(sat_decile = ntile(sat,n = 10)) %>% # bin the SAT score into 10 equally sized bins
  group_by(sat_decile,legacy) %>%
  summarise(prob_attend = mean(yield)) %>% # calculate average yield by sat bin and legacy
  ungroup()
```

```
## `summarise()` has grouped output by 'sat_decile'. You can override using the
## `.groups` argument.
```

```
toplot %>%
  ggplot(aes(x = factor(legacy),y = factor(sat_decile),fill = prob_attend)) +
  geom_tile() +
  scale_fill_gradient(low = "gray80",high = 'darkred')
```



Conditional Means

```
ad <- ad %>%
  mutate(sat_decile = ntile(sat,n= 10)) %>%
  group_by(sat_decile,legacy) %>%
  mutate(prob_attend = mean(yield)) %>%
  ungroup() %>%
  # select(sat_decile,legacy,yield,prob_attend) %>%
  mutate(pred_attend = ifelse(prob_attend > .5,1,0))

ad %>%
  select(yield,pred_attend,prob_attend)
```

```
## # A tibble: 2,150 × 3
##    yield pred_attend prob_attend
##    <int>       <dbl>       <dbl>
##  1     1           1       0.609
##  2     1           1       0.937
##  3     1           0       0.390
##  4     0           0       0.412
##  5     1           1       0.774
##  6     0           0       0.390
##  7     1           1       0.794
##  8     1           1       0.797
##  9     0           0       0.411
## 10     0           1       0.526
## # … with 2,140 more rows
```

Calculate Sensitivity & Specificity

```
ad %>%
  group_by(yield) %>%
  mutate(total_attend = n()) %>% # calculate total students who did (1) and did not (0) attend
  group_by(yield,total_attend,pred_attend) %>%
  summarise(nStudents = n()) %>% # Calculate total students by yield and by prediction
  mutate(proportion = nStudents / total_attend) # Transform into proportion
```

```
## `summarise()` has grouped output by 'yield', 'total_attend'. You can override
## using the `.groups` argument.
```

```
## # A tibble: 4 × 5
## # Groups:   yield, total_attend [2]
##   yield total_attend pred_attend nStudents proportion
##   <int>        <int>       <dbl>     <int>      <dbl>
## 1     0          684           0       304      0.444
## 2     0          684           1       380      0.556
## 3     1         1466           0       210      0.143
## 4     1         1466           1      1256      0.857
```

Linear Regression Model

```
mLM <- lm(yield ~ sat + net_price + legacy,ad)
summary(mLM)
```

```
##
## Call:
## lm(formula = yield ~ sat + net_price + legacy, data = ad)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1497 -0.3714  0.1338  0.3055  0.9392
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.981e+00  1.514e-01 -19.696  < 2e-16 ***
## sat          2.842e-03  1.173e-04  24.222  < 2e-16 ***
## net_price    1.052e-05  7.447e-07  14.122  < 2e-16 ***
## legacy       9.502e-02  1.954e-02   4.863 1.24e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.409 on 2146 degrees of freedom
## Multiple R-squared:  0.2302, Adjusted R-squared:  0.2291
## F-statistic: 213.9 on 3 and 2146 DF,  p-value: < 2.2e-16
```

```
ad %>%
  mutate(prob_attend = predict(mLM)) %>% # Predict regression to get probabilities
  mutate(pred_attend = ifelse(prob_attend > .5,1,0)) %>% # Convert probabilities to 0's & 1's
  group_by(yield) %>%
  mutate(total_attend = n()) %>% # Calculate total students who did and did not attend
  group_by(yield,pred_attend,total_attend) %>%
  summarise(nStudents = n()) %>% # Calculate total students who were accurately & inaccurately predicted
  ungroup() %>%
  mutate(proportion = nStudents / total_attend) # Calculate proportion
```

```
## `summarise()` has grouped output by 'yield', 'pred_attend'. You can override
## using the `.groups` argument.
```

```
## # A tibble: 4 × 5
##   yield pred_attend total_attend nStudents proportion
##   <int>       <dbl>        <int>     <int>      <dbl>
## 1     0           0          684       282     0.412
## 2     0           1          684       402     0.588
## 3     1           0         1466       113     0.0771
## 4     1           1         1466      1353     0.923
```

```
(282 + 1353) / 2150
```

```
## [1] 0.7604651
```

Looping over thresholds

```
threshRes <- NULL
for(thresh in seq(0,1,by = .025)) {
  tmp <- ad %>%
    mutate(pred_attend = ifelse(predict(mLM) > thresh,1,0)) %>%
    group_by(yield) %>%
    mutate(total_attend = n()) %>%
    group_by(yield,pred_attend,total_attend) %>%
    summarise(nStudents = n(),.groups = 'drop') %>%
    mutate(proportion = nStudents / total_attend) %>%
    mutate(threshold = thresh)

  threshRes <- threshRes %>%
    bind_rows(tmp)
}

threshRes
```

```
## # A tibble: 161 × 6
##    yield pred_attend total_attend nStudents proportion threshold
##    <int>       <dbl>        <int>     <int>      <dbl>     <dbl>
##  1     0           0          684         3    0.00439     0
##  2     0           1          684       681    0.996       0
##  3     1           1         1466      1466    1           0
##  4     0           0          684         5    0.00731     0.025
##  5     0           1          684       679    0.993       0.025
##  6     1           1         1466      1466    1           0.025
##  7     0           0          684         8    0.0117      0.05
##  8     0           1          684       676    0.988       0.05
##  9     1           1         1466      1466    1           0.05
## 10     0           0          684        12    0.0175      0.075
## # … with 151 more rows
```