Classification

Part 1

Prof. Bisbee

Vanderbilt University

Lecture Date: 2023/03/20

Slides Updated: 2023-03-19

Agenda

1. Classification
2. College Admissions

Definitions

- *Classification:* predicting the **class** of given data points via **predictive modeling**
    - *Class*: AKA targets, labels, or categories
    - *Predictive Modeling*: Approximate mapping function $f : X \rightarrow Y$
    - X: predictor variables
    - Y: outcome variable
    - f: ??

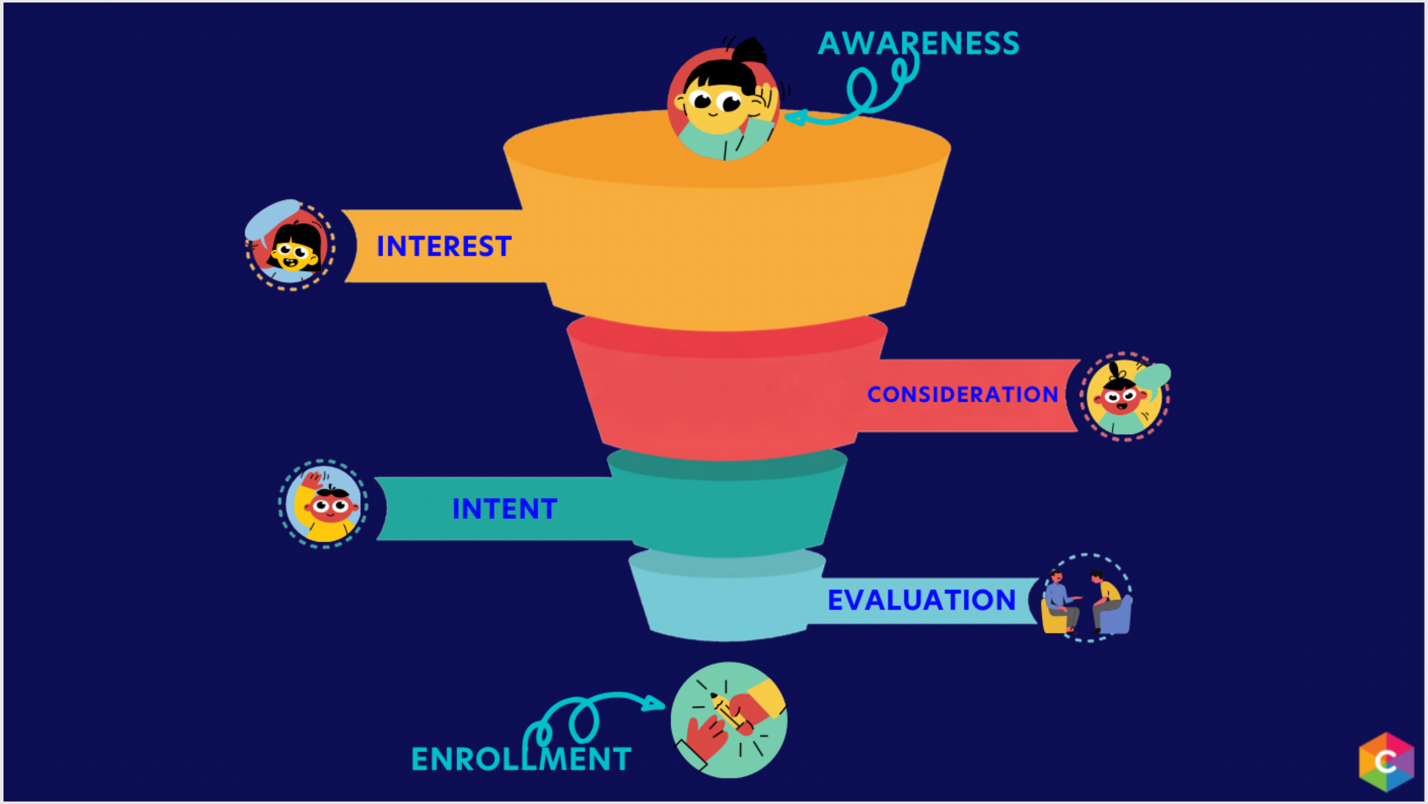
Mapping Functions

- We have already used a mapping functions!
- Linear Regression
  - f: $Y = \alpha + \beta X + \varepsilon$
- Underlying idea: X contain information about Y

It is in the Y

- If Y is continuous, we use OLS regression
- If Y is **binary**, we use "logistic" regression (AKA "logit")
  - As always, this is a **deep** area of study for those interested
- Today, using OLS for binary Y
  - Next few classes: replacing OLS regression with logit

College Admissions



- A live interactive infographic

College Admissions

- The math of college admissions

1. **Tuition** ($)
    - How they stay in business
2. **Reputation**
    - Higher reputation → more **tuition**
    - Based on academic qualifications

Data Science!

- This is a big industry for data scientists!
- Why?
    - If you screw this up, you lose A LOT OF MONEY
    - Too few students → not enough money to operate
    - Too many students → not enough capacity → bad reputation → not enough money
- Thus, we need people who are good at **classification**

Our Task

- Colleges hire data scientists to do more than just predict yield
- College **goals**: Increase reputation
  - Increase average SAT score to 1300
  - Admit at least 200 more students with incomes under $50,000
- College **constraints**: Stay in operation!
  - Maintain total revenues of $30m
  - Maintain entering class size of 1,500

How do we do this?

- Tuition discounting / targeting
  - Incentivize certain students to enroll
  - Make it cheaper for them to attend via **need-based** and **merit-based** aid
- **Need-based aid**:

$$need_{aid} = 500 + (income/1000 - 100) * -425$$

  - For every $1,000 less than $100,000, student receives +$425
- **Merit-based aid**:

$$merit_{aid} = 5000 + (sat/1001500)$$

  - For every 10 points in SAT scores above 1250, student receives extra $1,500

So how do we do this?

- Use tuition discounting to attract certain students
  - Those with higher SAT scores
  - Those with lower incomes
- Could give aid to everyone who fits these criteria
- But this is inefficient! Giving money to those would might not attend
- Want to **target** the aid toward those most likely to attend
- Again...**prediction**

Ethics

- Is this **ethical**?
- Ethics in data science is crucial

> [J]ust as the invention of the telescope revolutionized the study of the heavens, so too by **rendering the unmeasurable measurable**, the technological revolution in mobile, Web, and Internet communications has the potential to revolutionize our understanding of ourselves and how we interact.
>
> -- Duncan Watts (2011, p. 266)

- We will return to this topic in our final meeting

The Data

```
library(tidyverse)
library(scales)
ad<-read_rds("../data/admit_data.rds")%>%ungroup()
glimpse(ad)
```

```
## Rows: 2,150
## Columns: 14
## $ ID          <chr> "0001", "0002", "0003", "0004", "0005"…
## $ income      <dbl> 289720.59, 176763.29, 81204.02, 93320.…
## $ sat         <dbl> 1107.403, 1387.607, 1000.000, 1134.883…
## $ gpa         <dbl> 3.597153, 4.000000, 3.072323, 3.682776…
## $ visit       <dbl> 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,…
## $ legacy      <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
## $ registered  <dbl> 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0,…
## $ sent_scores <dbl> 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,…
## $ distance    <dbl> 10.23279, 89.75984, 152.29961, 317.502…
## $ tuition     <dbl> 45000, 45000, 45000, 45000, 45000, 450…
## $ need_aid    <dbl> 0.000, 0.000, 8488.293, 3338.779, 0.00…
## $ merit_aid   <dbl> 0.00, 35190.18, 0.00, 0.00, 30567.16, …
## $ net_price   <dbl> 45000.000, 9809.815, 36511.707, 41661.…
## $ yield       <int> 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0,…
```

The Data

- Start with the basics:

  1. What is the unit of analysis?

  2. Which variables are we interested in?

Prediction

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots + \varepsilon$$

- Y: enrollment (`yield`)
- X: ??
  - In prediction, we don't care about **theory** or **research questions**
  - Just want to maximize **accuracy**...which X's are the "best"?
- Look at univariate & conditional relationships

The Data

- Outcome Y: `yield`

```
ad %>%
  summarise(`Yield Rate` = percent(mean(yield)))
```

```
##   Yield Rate
## 1        68%
```

- Multivariate analysis?

Which X?

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

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

Which X?

```
ad %>%
  group_by(visit) %>%
  summarise(pr_attend = mean(yield))
```

```
## # A tibble: 2 × 2
##   visit pr_attend
##   <dbl>     <dbl>
## 1     0     0.644
## 2     1     0.736
```

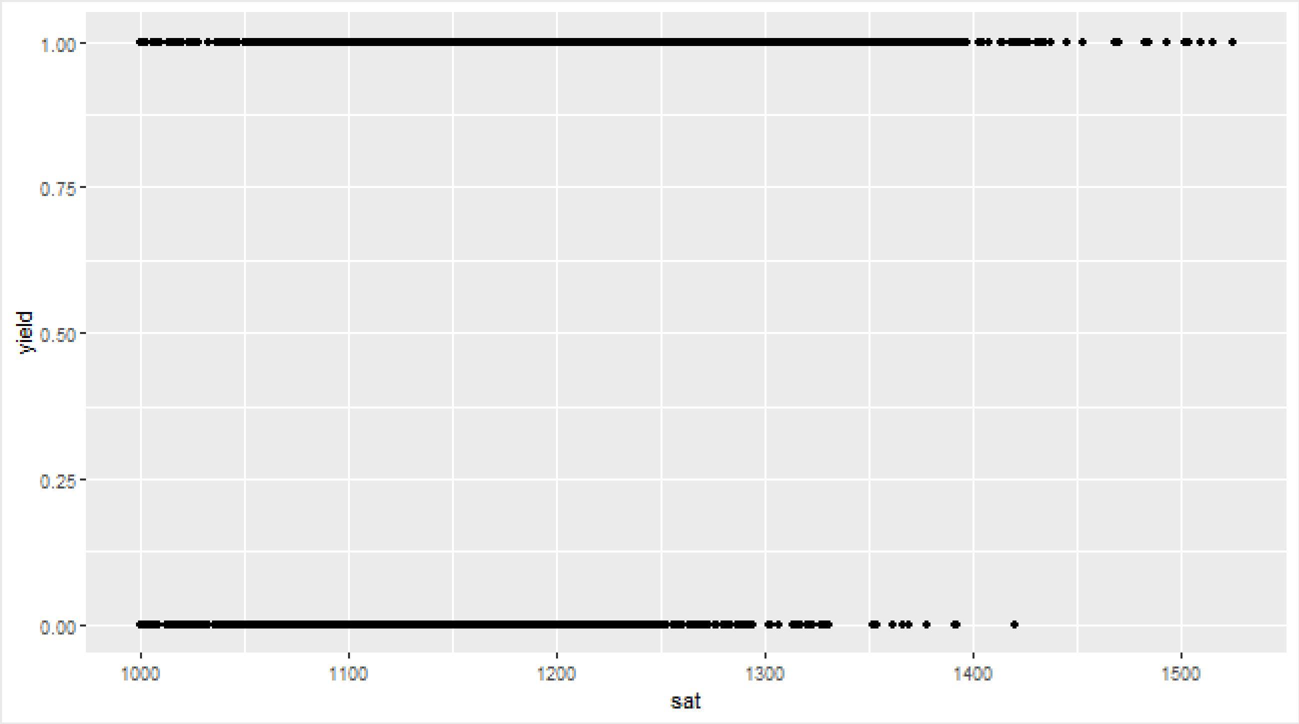
Which X?

```r
ad %>%
  group_by(sent_scores) %>%
  summarise(pr_attend = mean(yield))
```

```
## # A tibble: 2 × 2
##   sent_scores pr_attend
##         <dbl>     <dbl>
## 1           0     0.651
## 2           1     0.805
```

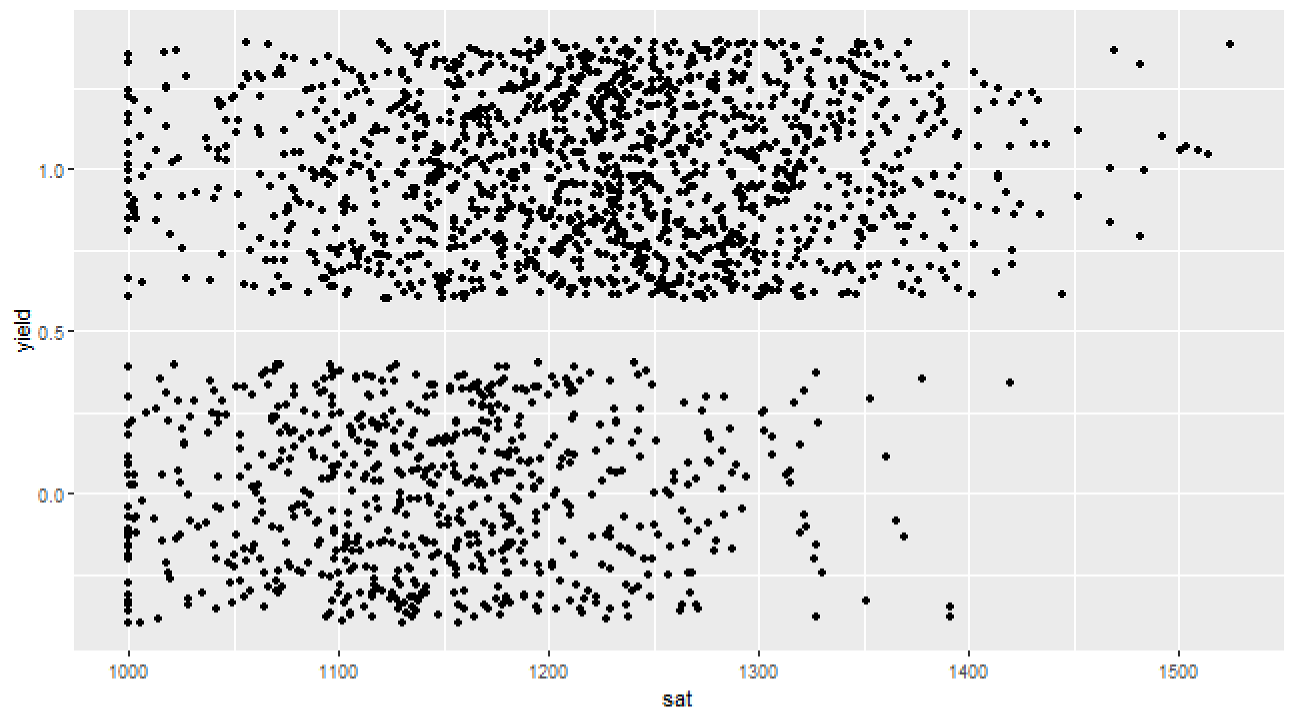
Which X?

```
ad %>%
  ggplot(aes(x = sat,y = yield)) +
  geom_point()
```



Which X?

```
ad %>%
  ggplot(aes(x = sat,y = yield)) +
  geom_jitter()
```



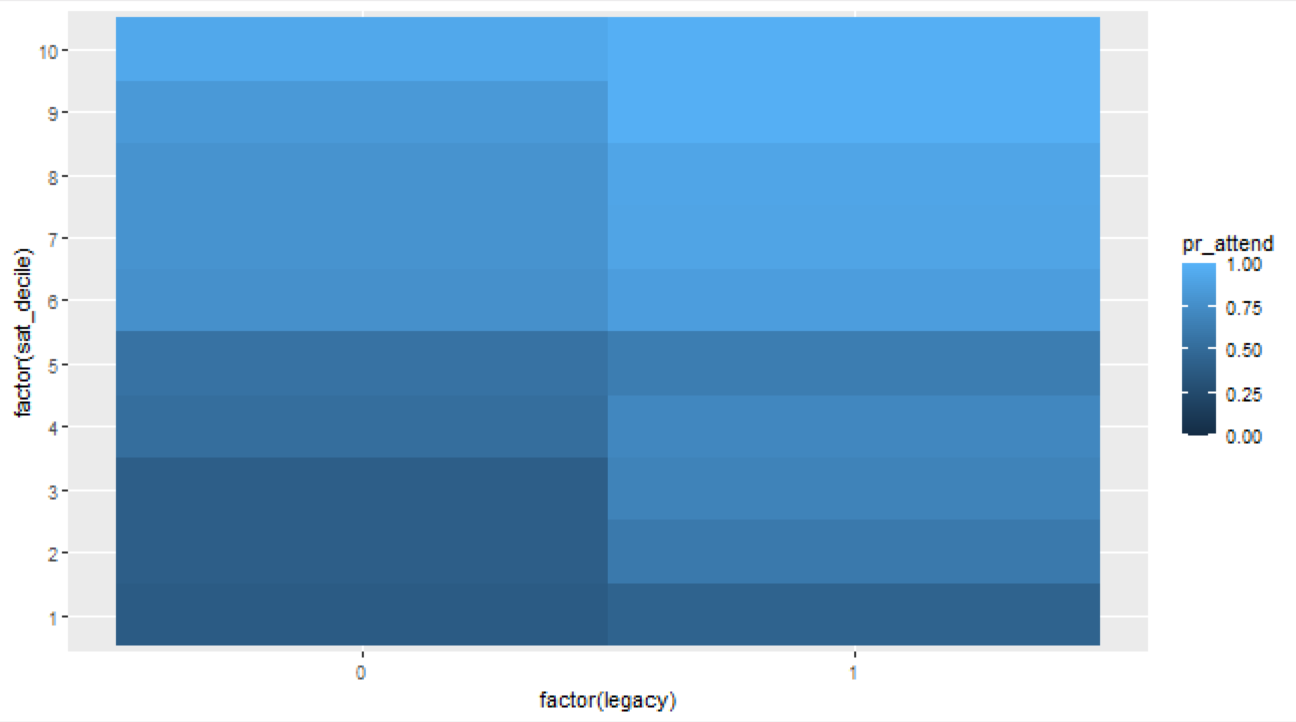
Heatmaps

- Look at 3-dimensions of data
    - Done this before by tweaking `fill`, `color`, or `size`
- `geom_tile()`: create a heatmap

```
p <- ad %>%
  mutate(sat_decile = ntile(sat,n=10)) %>% # Bin SAT by decile (10%)
  group_by(sat_decile,legacy) %>% # Calculate average yield by SAT &
legacy
  summarise(pr_attend = mean(yield),
            .groups = 'drop') %>%
  ggplot(aes(x = factor(legacy),y = factor(sat_decile), # Both x and
y-axes are factors
             fill = pr_attend)) + # Fill by third dimension
  geom_tile() + # Creates rectangles
  scale_fill_gradient(limits = c(0,1)) # Set fill color (can do much
more here)
```

Heatmaps

```
p
```



10
9
8
7
6
5
4
3
2
1
factor(sat_decile)
0
1
factor(legacy)
pr_attend
1.00
0.75
0.50
0.25
0.00

Simplest Predictions

- Remember: regression is just fancier conditional means

```
ad <- ad %>%
  mutate(sat_decile = ntile(sat,n=10)) %>% # Bin SAT by decile (10%)
  group_by(sat_decile,legacy) %>% # Calculate average yield by SAT & legacy
  mutate(prob_attend = mean(yield)) %>% # use mutate() instead of summarise() to avoid collapsing the data
  mutate(pred_attend = ifelse(prob_attend > .5,1,0)) %>% # If the probability is greater than 50-50, predict they attend
  ungroup()
```

Simplest Predictions

- Conditional means

```r
ad %>%
  group_by(yield,pred_attend) %>%
  summarise(nStudents=n(),.groups = 'drop')
```

```
## # A tibble: 4 × 3
##   yield pred_attend nStudents
##   <int>       <dbl>     <int>
## 1     0           0       304
## 2     0           1       380
## 3     1           0       210
## 4     1           1      1256
```

Accuracy

- What is "accuracy"?
  - Proportion "correct" predictions
- For a binary outcome, "accuracy" has two dimensions
  - Proportion of correct `1`s: **Sensitivity**
  - Proportion of correct `0`s: **Specificity**

Accuracy

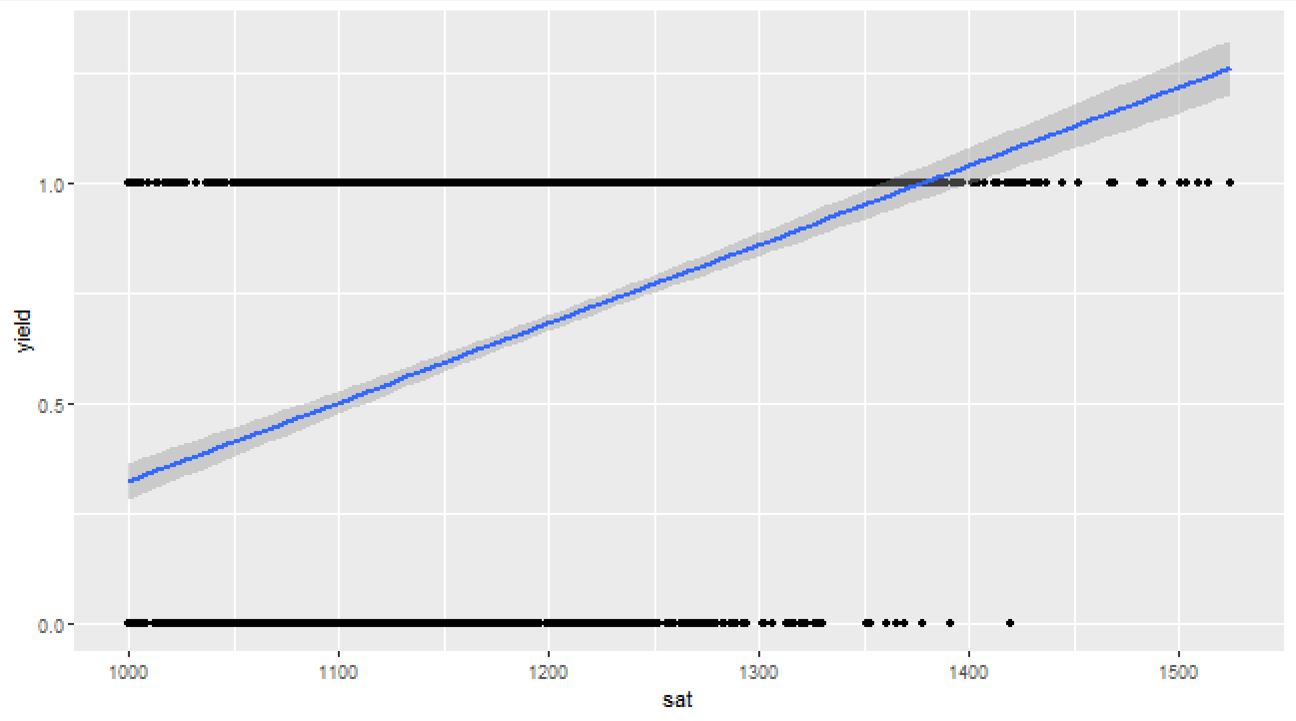
```
ad %>%
  group_by(yield) %>%
  mutate(total_attend = n()) %>%
  group_by(yield,pred_attend,total_attend) %>%
  summarise(nStudents=n(),.groups = 'drop') %>%
  mutate(prop = nStudents / total_attend)
```

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

- Overall accuracy: `(304 + 1256) / 2150` = 73%

Regression

```
ad %>%
  ggplot(aes(x = sat,y = yield)) +
  geom_point() +
  geom_smooth(method = 'lm')
```



Regression

- Binary outcome variable!
  - A linear regression is not the best solution
  - Predictions can exceed support of Y
- But it can still work! **linear probability model**

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

Linear Regression

```
require(broom) # broom package makes it easy to read regression
output
```

```
## Loading required package: broom
```

```
tidy(mLM) %>% # This would be the same as summary(mLM)
  mutate_at(vars(-term),function(x) round(x,5))
```

```
## # A tibble: 4 × 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept) -2.98      0.151      -19.7        0
## 2 sat          0.00284   0.00012     24.2        0
## 3 net_price    0.00001   0           14.1        0
## 4 legacy       0.0950    0.0195       4.86       0
```

Linear Regression

```r
mLM <- lm(yield ~ scale(sat) + scale(net_price) + legacy,ad)
tidy(mLM)
```

```
## # A tibble: 4 × 5
##   term             estimate std.error statistic   p.value
##   <chr>               <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)        0.654     0.0105     62.3  0
## 2 scale(sat)         0.280     0.0116     24.2  8.84e-115
## 3 scale(net_price)   0.164     0.0116     14.1  2.30e- 43
## 4 legacy             0.0950    0.0195      4.86 1.24e-  6
```

```r
ad %>%
  summarise_at(vars(sat,net_price),function(x) round(sd(x),1))
```

```
## # A tibble: 1 × 2
##     sat net_price
##   <dbl>     <dbl>
## 1  98.6    15569.
```

Evaluating Predictions

```
ad %>%
  mutate(preds = predict(mLM)) %>%
  mutate(predBinary = ifelse(preds > .5,1,0)) %>%
  select(yield,predBinary,preds)
```

```
## # A tibble: 2,150 × 3
##    yield predBinary preds
##    <int>      <dbl> <dbl>
##  1     1          1 0.735
##  2     1          1 1.07
##  3     1          0 0.245
##  4     0          1 0.683
##  5     1          1 0.589
##  6     0          0 0.358
##  7     1          1 0.559
##  8     1          1 0.757
##  9     0          0 0.366
## 10     0          1 0.698
## # … with 2,140 more rows
```

Evaluating Predictions

```
ad %>%
  mutate(pred_attend = ifelse(predict(mLM) > .5,1,0)) %>%
  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 = percent(sum((yield == pred_attend)*nStudents) /
sum(nStudents)))
```

```
## # A tibble: 4 × 6
##   yield pred_attend total_attend nStudents   prop accuracy
##   <int>       <dbl>        <int>     <int>  <dbl> <chr>
## 1     0           0          684       282 0.412  76%
## 2     0           1          684       402 0.588  76%
## 3     1           0         1466       113 0.0771 76%
## 4     1           1         1466      1353 0.923  76%
```

Evaluating Predictions

- Overall accuracy is just the number of correct predictions (either `0` or `1`) out of all possible
    - Is 76% good?
    - What would the dumbest guess be? Everyone will attend! 68%
- Might also want to care about just `1`s
    - **Sensitivity**: Predicted attendees / actual attendees = 92.3%
- Also might care about just `0`s
    - **Specificity**: Predicted non-attendees / actual non-attendees = 41.2%

Thresholds

- Shifting the threshold for `0` or `1` prediction can matter

```
ad %>%
  mutate(pred_attend = ifelse(predict(mLM) > .4,1,0)) %>%
  group_by(yield) %>%
  mutate(total_attend = n()) %>%
  group_by(yield,pred_attend,total_attend) %>%
  summarise(nStudents=n(),.groups = 'drop') %>%
  mutate(prop = percent(nStudents / total_attend)) %>%
  ungroup() %>%
  mutate(accuracy = percent(sum((yield == pred_attend)*nStudents) /
sum(nStudents)))
```

```
## # A tibble: 4 × 6
##   yield pred_attend total_attend nStudents prop  accuracy
##   <int>       <dbl>        <int>     <int> <chr> <chr>
## 1     0           0          684       176 26%   74%
## 2     0           1          684       508 74%   74%
## 3     1           0         1466        58 4%    74%
## 4     1           1         1466      1408 96%   74%
```

Thresholds

- Shifting the threshold for `0` or `1` prediction can matter

```
ad %>%
  mutate(pred_attend = ifelse(predict(mLM) > 1,1,0)) %>%
  group_by(yield) %>%
  mutate(total_attend = n()) %>%
  group_by(yield,pred_attend,total_attend) %>%
  summarise(nStudents=n(),.groups = 'drop') %>%
  mutate(prop = percent(nStudents / total_attend)) %>%
  ungroup() %>%
  mutate(accuracy = percent(sum((yield == pred_attend)*nStudents) /
sum(nStudents)))
```

```
## # A tibble: 4 × 6
##   yield pred_attend total_attend nStudents prop  accuracy
##   <int>       <dbl>        <int>     <int> <chr> <chr>
## 1     0           0          684       683 99.9% 38%
## 2     0           1          684         1 0.1%  38%
## 3     1           0         1466      1342 91.5% 38%
## 4     1           1         1466       124 8.5%  38%
```

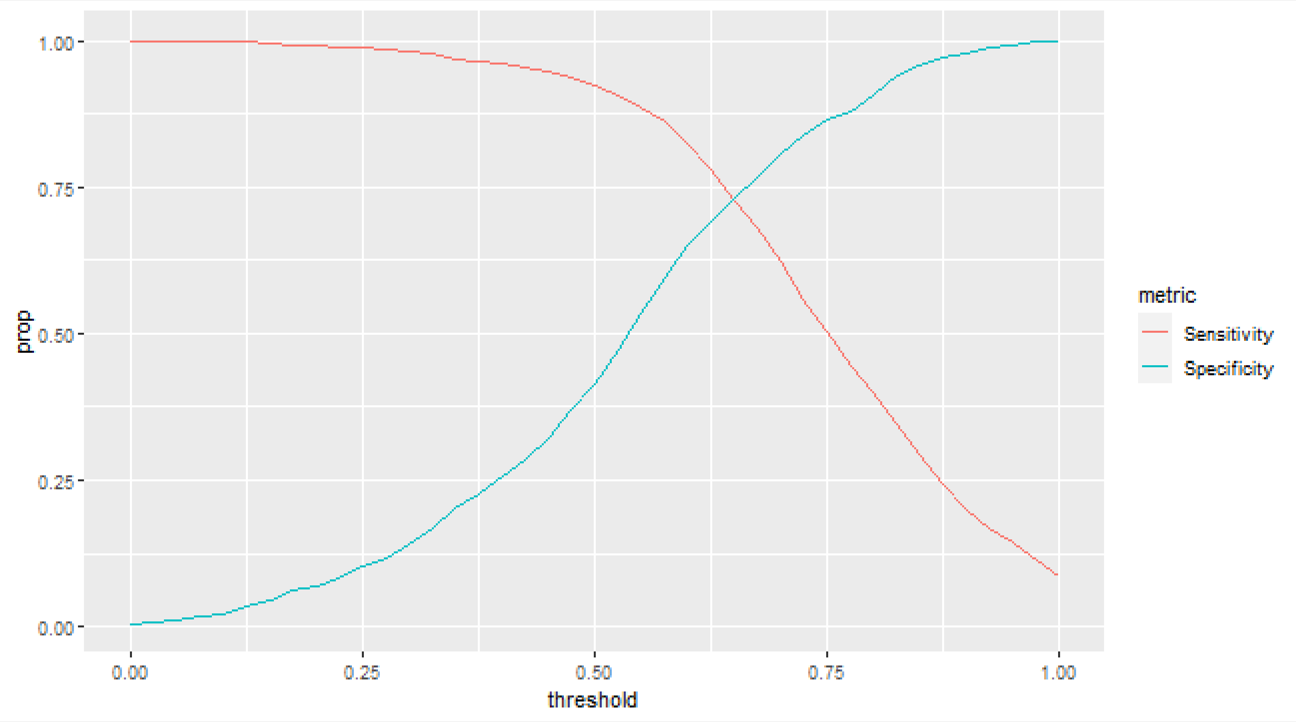
Thresholds

- Let's loop it!

```
toplot <- NULL
for(thresh in seq(0,1,by = .025)) {
  toplot <- 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(prop = nStudents / total_attend) %>%
  ungroup() %>%
  mutate(accuracy = sum((yield == pred_attend)*nStudents) /
sum(nStudents)) %>%
  mutate(threshold = thresh) %>%
    bind_rows(toplot)
}
```

Thresholds

```
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()
```



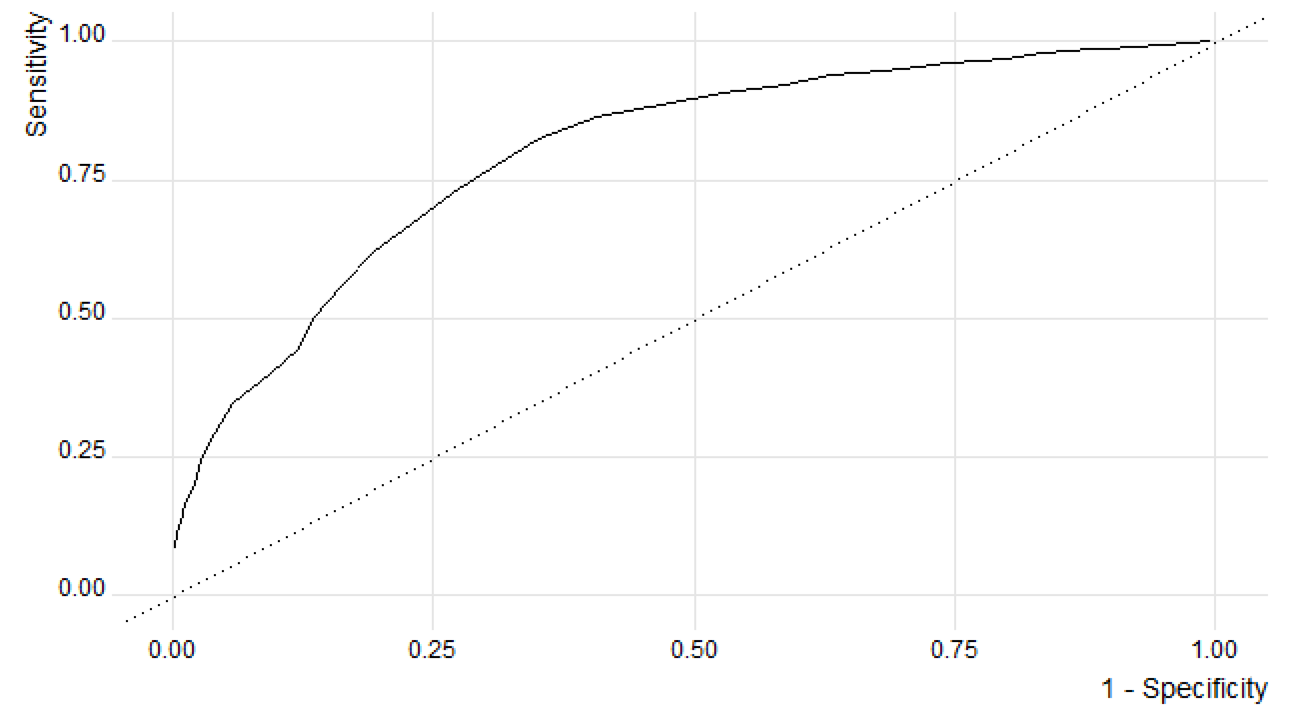
ROC Curve

- Receiver-Operator Characteristic (ROC) Curve
- Commonly used to evaluate classification methods
    - X-axis: 1-specificity
    - Y-axis: sensitivity

```
p <- toplot %>%
  mutate(metric = ifelse(yield == 1 & pred_attend == 1,'Sensitivity',
                         ifelse(yield == 0 & pred_attend ==
0,'Specificity',NA))) %>%
  drop_na(metric) %>%
  select(prop,metric,threshold) %>%
  spread(metric,prop) %>%
  ggplot(aes(x = 1-Specificity,y = Sensitivity)) +
  geom_line() +
  xlim(c(0,1)) + ylim(c(0,1)) +
  geom_abline(slope = 1,intercept = 0,linetype = 'dotted') +
  ggridges::theme_ridges()
```

ROC Curve

```
p
```



- Better models have high levels of sensitivity **and** specificity at every threshold

AUC Measure

- Area Under the Curve (AUC)
  - A single number summarizing classification performance

```
require(tidymodels)
roc_auc(data = ad %>%
  mutate(pred_attend = predict(mLM),
         truth = factor(yield,levels = c('1','0'))) %>%
  select(truth,pred_attend),truth,pred_attend)
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.801
```

Party time!

- Adding more variables / trying different combinations
- **Workflow**
    1. Train models
    2. Predict models
    3. Evaluate models

Train models

```
m1 <- lm(yield ~ sat + net_price + legacy,ad)
m2 <- lm(yield ~ sat + net_price + legacy + income,ad)
m3 <- lm(yield ~ sat + net_price + legacy + income + gpa,ad)
m4 <- lm(yield ~ sat + net_price + legacy + income + gpa +
distance,ad)
m5 <- lm(yield ~ sat + net_price + legacy + income + gpa + distance +
visit,ad)
m6 <- lm(yield ~ sat + net_price + legacy + income + gpa + distance +
visit + registered + sent_scores,ad)
```

Predict models

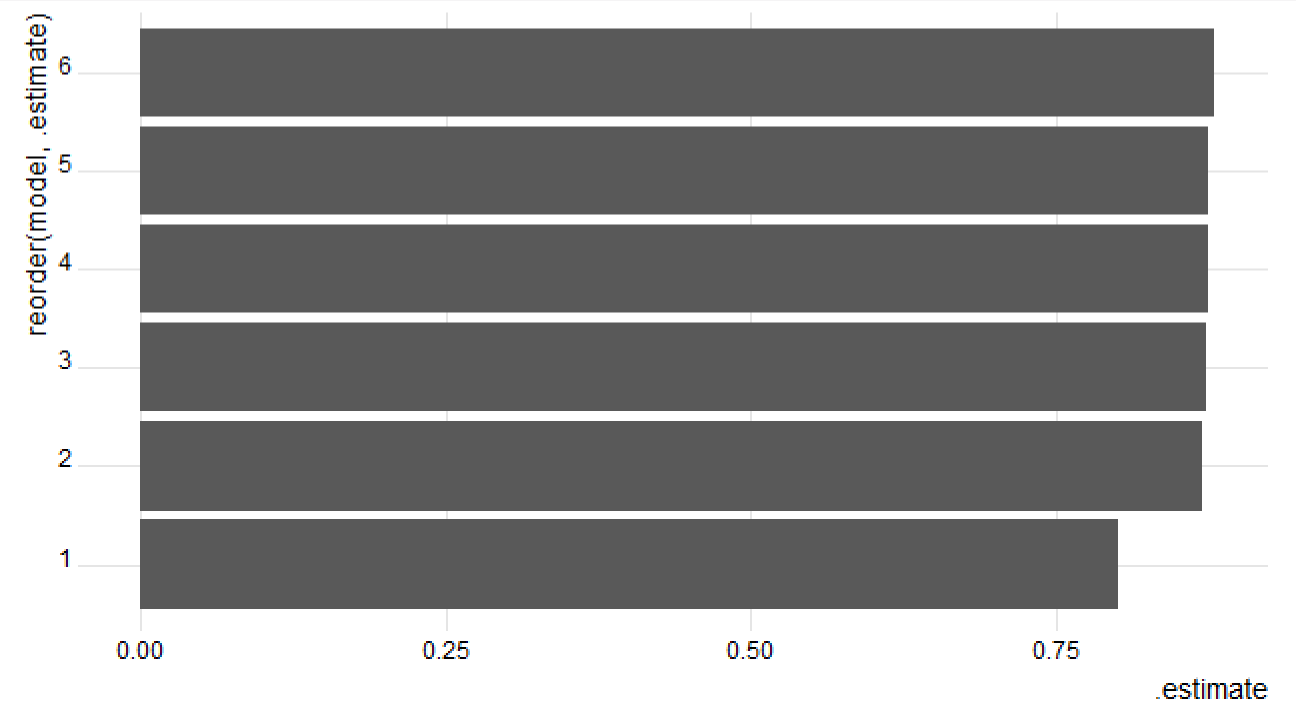
```
toEval <- ad %>%
  mutate(m1Preds = predict(m1),
         m2Preds = predict(m2),
         m3Preds = predict(m3),
         m4Preds = predict(m4),
         m5Preds = predict(m5),
         m6Preds = predict(m6),
         truth = factor(yield,levels = c('1','0')))
```

Evaluate models

```r
rocRes <- NULL
for(model in 1:6) {
  rocRes <- roc_auc(toEval,truth,paste0('m',model,'Preds')) %>%
    mutate(model = model) %>%
    bind_rows(rocRes)
}
```

Evaluate models

```
rocRes %>%
  ggplot(aes(x = .estimate,y = reorder(model,.estimate))) +
  geom_bar(stat = 'identity') +
  ggridges::theme_ridges()
```



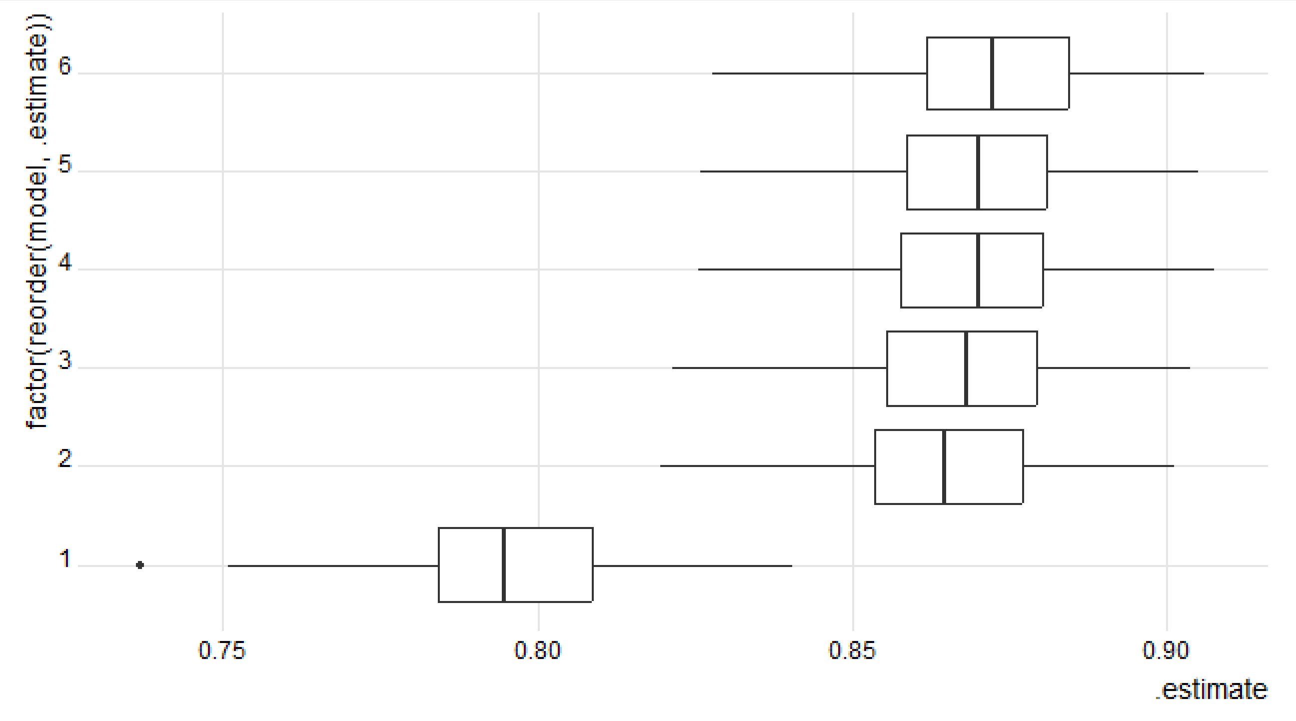
OVERFITTING

- Cross validation to the rescue!

```
set.seed(123)
cvRes <- NULL
for(i in 1:100) {
  # Cross validation prep
  inds <- sample(1:nrow(ad),size = round(nrow(ad)*.8),replace = F)
  train <- ad %>% slice(inds)
  test <- ad %>% slice(-inds)

  # Training models
  m1 <- lm(yield ~ sat + net_price + legacy,train)
  m2 <- lm(yield ~ sat + net_price + legacy + income,train)
  m3 <- lm(yield ~ sat + net_price + legacy + income + gpa,train)
  m4 <- lm(yield ~ sat + net_price + legacy + income + gpa + distance,train)
  m5 <- lm(yield ~ sat + net_price + legacy + income + gpa + distance + visit,train)
  m6 <- lm(yield ~ sat + net_price + legacy + income + gpa + distance + visit + registered + sent_scores,train)

  # Predicting models
  toEval <- test %>%
    mutate(m1Preds = predict(m1,newdata = test),
           m2Preds = predict(m2,newdata = test),
           m3Preds = predict(m3,newdata = test),
           m4Preds = predict(m4,newdata = test),
           m5Preds = predict(m5,newdata = test),
           m6Preds = predict(m6,newdata = test),
           truth = factor(yield,levels = c('1','0')))

  # Evaluating models
  rocRes <- NULL
  for(model in 1:6) {
    rocRes <- roc_auc(toEval,truth,paste0('m',model,'Preds')) %>%
      mutate(model = model) %>%
      bind_rows(rocRes)
  }
  cvRes <- rocRes %>%
    mutate(bsInd = i) %>%
    bind_rows(cvRes)
}
```

Cross Validation AUC

```
cvRes %>%
  ggplot(aes(x = .estimate,y = factor(reorder(model,.estimate)))) +
  geom_boxplot() +
  ggridges::theme_ridges()
```



Conclusion

- Classification is just a type of prediction
  - We used linear regression
  - But there are **much** fancier algorithms out there
- Next class:
  - A *slightly* fancier algorithm: logistic regression
  - How to use the models to achieve the university's goals
- Go to Brightspace and take the **12th** quiz
  - The password to take the quiz is ####
- **Homework:**
  - Problem Set 6 (due 2023-03-24 by 11:59PM)