College Admissions, Parts 1 & 2

Homework

Prof. Bisbee

Due Date: 2023-10-30 & 11-01

College Admissions: From the College's View

All of you have quite recently gone through the stressful process of figuring out which college to attend. You most likely selected colleges you thought might be a good fit, sent off applications, heard back from them, and then weighed your options. Those around you probably emphasized what an important decision this is for you and for your future.

Colleges see this process from an entirely different point of view. A college needs students to enroll first of all in order to collect enough tuition revenues in order to keep the lights on and the faculty paid. Almost all private colleges receive most of their revenues from tuition, and public colleges receive about equal amounts of funding from tuition and state funds, with state funds based on how many students they enroll. Second, colleges want to enroll certain types of students– colleges based their reputation based on which students enroll, with greater prestige associated with enrolling students with better demonstrated academic qualifications. The job of enrolling a class that provides enough revenue AND has certain characteristics falls to the Enrollment Management office on a campus. This office typically includes the admissions office as well as the financial aid office.

The College Admissions "Funnel"

The admissions funnel (https://www.curacubby.com/blog/admissions-funnel) is a well-established metaphor for understanding the enrollment process from the college's perspective. It begins with colleges identifying prospective students: those who might be interested in enrolling. This proceeds to "interested" students, who engage with the college via registering on the college website, sending test scores, visiting campus, or requesting other information. Some portion of these interested students will then apply. Applicants are then considered, and admissions decisions are made. From this group of admitted students a certain proportion will actually enroll. Here's live data from UC Santa Cruz (go Banana Slugs!) on their enrollment funnel (https://iraps.ucsc.edu/iraps-public-dashboards/student-demand/admissions-funnel.html).

Each stage in this process involves modeling and prediction: how can we predict which prospective students will end up being interested students? How many interested students will turn into applicants? And, most importantly, how many admitted students will actually show up in the fall?

Colleges aren't powerless in this process. Instead, they execute a careful strategy to intervene at each stage to get both the number and type of students they want to convert to the next stage. These are the core functions of enrollment management. Why did you receive so many emails, brochures and maybe even text messages? Some model somewhere said that the intervention could convert you from a prospect to an interest, or from an interest to an applicant.

We're going to focus on the very last step: from admitted students to what's called a yield: a student who actually shows up and sits down for classes in the fall.

The stakes are large: if too few students show up, then the institutions will not have enough revenues to operate. If too many show up the institution will not have capacity for all of them. On top of this, enrollment managers are also tasked with the combined goals of increasing academic prestige (usually through test scores and GPA) and increasing the socioeconomic diversity of the entering class. As we'll see, these are not easy tasks.

The Data

We're going to be using a dataset that was constructed to resemble a typical admissions dataset. To be clear: this is not real data, but instead is based on the relationships we see in actual datasets. Using real data in this case would be a violation of privacy.

```
library(tidyverse)
# library(tidymodels)
library(scales)
```

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

Codebook for `admit_data.rds` :

Variable Name	Description
ID	Student id
income	Family income (AGI)
sat	SAT/ACT score (ACT scores converted to SAT scale)
gpa	HS GPA, four point scale
visit	Did student visit campus?
legacy	Did student parent go to this college?
registered	Did student register on the website?
sent_scores	Did student send scores prior to applying?
distance	Distance from student home address to campus
tuition	Stated tuition: $45,000
need_aid	Need-based aid offered
merit_aid	Merit-based aid offered
net_price	Net Price: Tuition less aid received
yield	Student enrolled in classes in fall after admission

The Basics

```
## How many admitted students enroll?
ad%>%summarize(`Yield Rate`=percent(mean(yield)))
```

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

```
## Just for enrolled students
ad%>%filter(yield==1)%>%summarize(
               `Average  SAT Score`=number(mean(sat),accuracy=1,big.mark=""),
               `Average GPA`=number(mean(gpa),accuracy = 1.11),
               `Tuition`=dollar(mean(tuition)),
               `Average Net Price`=dollar(mean(net_price),accuracy = 1 ),
               `Total Tuition Revenues`=dollar(sum(net_price)),
               `Total 1st Year Enrollment`=comma(n(),big.mark=","))
```

```
##   Average  SAT Score Average GPA Tuition Average Net Price
## 1               1226        3.33 $45,000           $20,924
##   Total Tuition Revenues Total 1st Year Enrollment
## 1            $30,674,149                     1,466
```

So, a few things stand out right away, all of which are pretty common among private colleges. First, this is a moderately selective institution, with an average GPA of 3.33 (unweighted) and an average SAT of about 1200 (about a 25 on the ACT). The average net price is MUCH less than tuition, indicating that the campus is discounting heavily. Total revenues from tuition are about 30 million.

The Case

We've been hired as the data science team for a liberal arts college this is a real thing (https://www.fire-engine-red.com/financial-aid-and-student-success/).

The college president and the board of trustees have two strategic goals:

1. Increase the average SAT score to 1300
2. Admit at least 200 more students with incomes less than $50,000

Here's the rub: they want to do this without allowing tuition revenues to drop below $30 million and without changing the size of the entering class, which should be about 1,500 students (plus or minus 50, nobody minds sleeping in the study lounge, right?).

What we need to do is to figure out which students are most and least likely to enroll. We can then target our financial aid strategy to improve yield rates among certain groups.

This is a well-known problem known as price discrimination (http://sites.bu.edu/manove-ec101/files/2017/11/VarianHalPriceDiscrimination1989.pdf), which is applied in many industries, including airlines, hotels, and software. The idea is to charge the customers who are most willing/able to pay the most, while charging the customers who are least willing/able to pay the least.

To solve our problem we need to do the following:

1. Come up with a prediction algorithm that accurately establishes the relationship between student characteristics and the probability of attendance
2. Adjust policies in order to target those students who we want to attend, thereby increasing their probability of attendance.

Current Institutional Policies

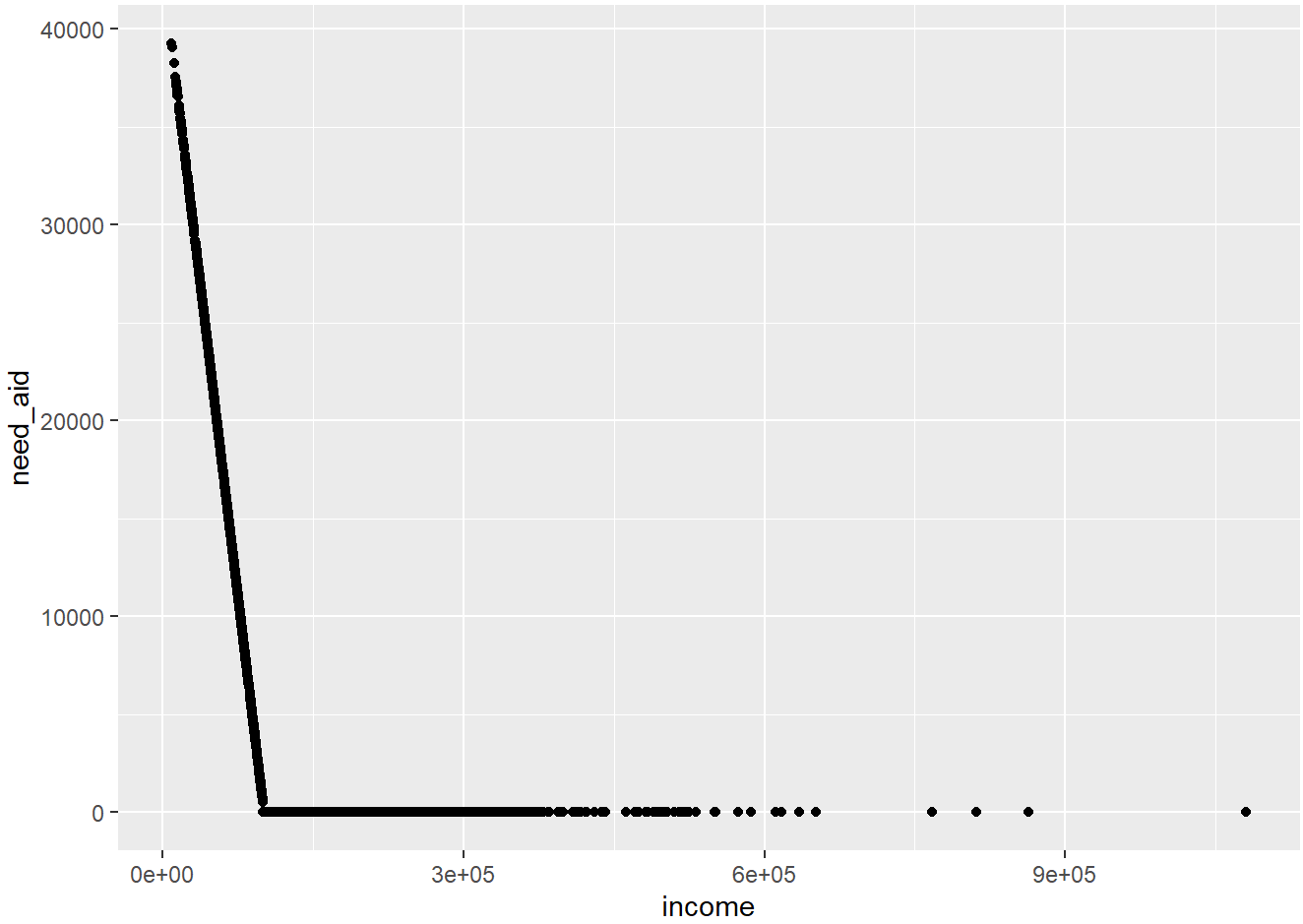
Essentially every private college engages heavily in tuition discounting. This has two basic forms: need-based aid and merit-based aid. Need-based aid is provided on the basis of income, typically with some kind of income cap. Merit-based aid is based on demonstrated academic qualifications, again usually with some kind of minimum. Here's this institution's current policies.

The institution is currently awarding need-based aid for families making less than $100,0000 on the following formula:

$$need_aid = 500 + (income/1000 - 100)* - 425$$

Translated, this means for every $1,000 the family makes less than $100,000 the student receives an additional 425 dollars. So for a family making $50,000, need-based aid will be
$500 + (50,000/1000 - 100) * -425 = 500 + (-50 * -425)$=$21,750. Need based aid is capped at total tuition.

```
ad%>%
  ggplot(aes(x=income,y=need_aid))+
  geom_point()
```

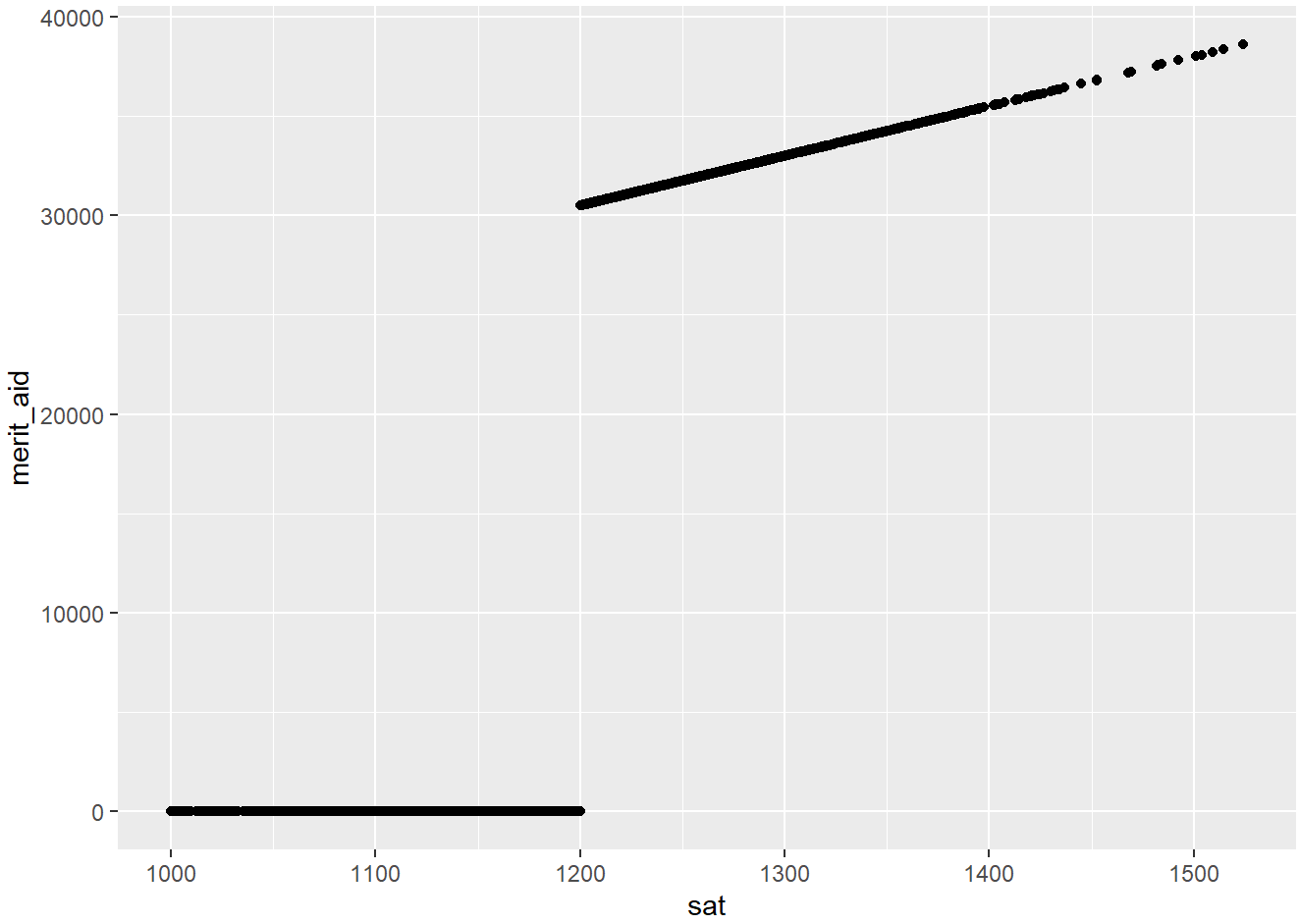


The institution is currently awarding merit-based aid for students with SAT scores above 1250 on the following formula:

$$merit_aid = 5000 + (sat/100*1500)$$

Translated, this means that for every 10 points in SAT scores above 1250, the student will receive an additional $1,500. So for a student with 1400 SAT, merit based aid will be : $5000 + (1400/10 * 250) = 500 + 140 * 250$

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



As with need-based aid, merit-based aid is capped by total tuition.

Classification

Our core prediction problem is classification (https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623). There are two groups of individuals that constitute our outcome: those who attended and those who did not. In data science, predicting group membership is known as a classification problem. It occurs whenever the outcome is a set of discrete groupings. We'll be working with the simplest type of classification problem, which has just two groups, but these problems can have multiple groups– essentially categorical variables.

Probability of Attendance

Remember: the mean of a binary variable is the same thing as the proportion of the sample with that characteristic. So, the mean of `yield` is the same thing as the proportion of admitted students who attend, or the probability of attendance.

```
ad%>%summarize(pr_attend=mean(yield))
```

```
##   pr_attend
## 1 0.6818605
```

Conditional Means

Let's go back to our first algorithm for prediction: conditional means. Let's start with the variable `legacy` which indicates whether or not the student has a parent who attended the same institution:

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

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

That's a big difference! Legacy students are abut 17 percentage points more likely to yield than non-legacies.

Next, let's look at SAT scores. This is a continuouos variable, so we need to first break it up into a smaller number of groups. Let's look at yield rates by quintile of SAT scores among admitted students:

```
ad%>%
  mutate(sat_quintile=ntile(sat,n=5))%>%
  group_by(sat_quintile)%>%
  summarize(min_sat=min(sat),
  pr_attend=mean(yield))
```

```
## # A tibble: 5 × 3
##   sat_quintile min_sat pr_attend
##          <int>   <dbl>     <dbl>
## 1            1   1000      0.440
## 2            2   1114.     0.533
## 3            3   1173.     0.691
## 4            4   1227.     0.828
## 5            5   1285.     0.919
```

So, it looks like yield steadily increases with SAT scores– a good sign for the institution as it seeks to increase SAT scores.

Quick Exercise calculate yield by quintiles of net price: what do you see?

```
ad%>%
  mutate(net_price_quintie=ntile(net_price,n=5))%>%
  group_by(net_price_quintie)%>%
  summarize(amount=min(net_price),
            pr_attend=mean(yield))
```

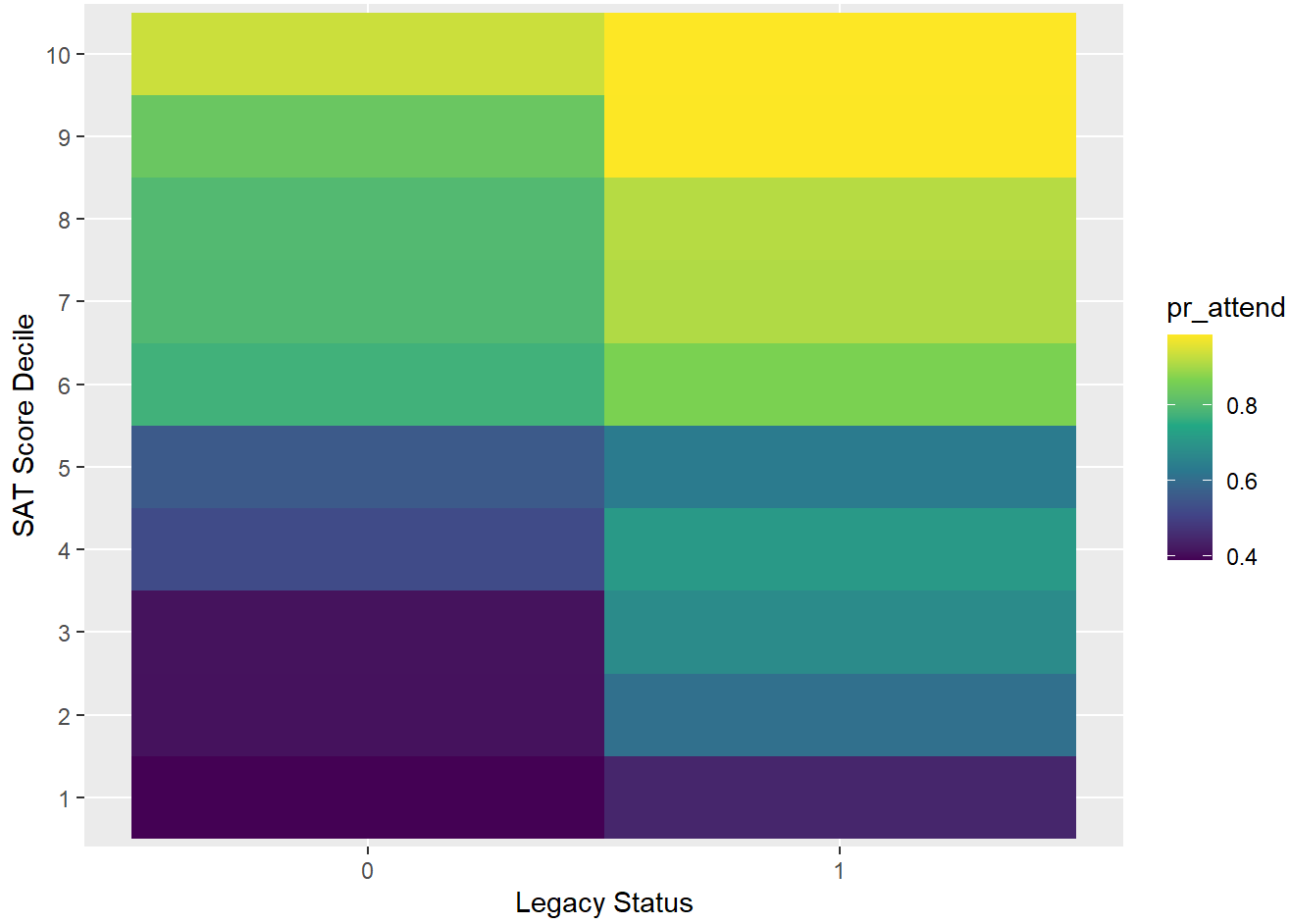
```
## # A tibble: 5 × 3
##   net_price_quintie amount pr_attend
##               <int>  <dbl>     <dbl>
## 1                 1     0      0.649
## 2                 2  9464.     0.902
## 3                 3 13053.     0.670
## 4                 4 22505.     0.349
## 5                 5 43766.     0.840
```

Combining Conditional Means

Let's look at yield rates by both sat quintile and legacy status.

```
ad%>%
  mutate(sat_decile=ntile(sat,n=10))%>%
  group_by(sat_decile,legacy)%>%
  summarize(min_sat=min(sat),
  pr_attend=mean(yield))%>%
  ggplot(aes(y=as_factor(sat_decile),x=as_factor(legacy),fill=pr_attend))+
  geom_tile()+
  scale_fill_viridis_c()+
  ylab("SAT Score Decile")+xlab("Legacy Status")
```

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



Predictions based on conditional means

We can use this simple method to make predictions.

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

Let's compare this predicted with the actual:

```
ad%>%
  group_by(yield,pred_attend)%>%
  summarize(n())%>%
  rename(`Actually Attended`=yield,
         `Predicted to Attend`=pred_attend,
         `Number of Students`=`n()`)
```

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

```
## # A tibble: 4 × 3
## # Groups:   Actually Attended [2]
##   `Actually Attended` `Predicted to Attend` `Number of Students`
##                 <int>                 <dbl>                 <int>
## 1                   0                     0                   304
## 2                   0                     1                   380
## 3                   1                     0                   210
## 4                   1                     1                  1256
```

Acccuracy of Conditional Means

```
ad%>%
  group_by(yield)%>%
  mutate(total_attend=n())%>%
  group_by(yield,pred_attend)%>%
  summarize(n(),`Actual Group`=mean(total_attend))%>%
  mutate(Proportion=`n()`/`Actual Group`)%>%
  rename(`Actually Attended`=yield,
         `Predicted to Attend`=pred_attend,
         `Number of Students`=`n()`)
```

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

```
## # A tibble: 4 × 5
## # Groups:   Actually Attended [2]
##   `Actually Attended` `Predicted to Attend` `Number of Students` `Actual Group`
##                 <int>                 <dbl>                 <int>          <dbl>
## 1                   0                     0                   304            684
## 2                   0                     1                   380            684
## 3                   1                     0                   210           1466
## 4                   1                     1                  1256           1466
## # ℹ 1 more variable: Proportion <dbl>
```

Here's how to read this: There were 312 students that our algorithm said would not attend who didn't attend. This means out of the 674 students who were admitted but did not attend, our algorithm correctly classified 46 percent. There were 362 students who our model said would not attend who actually showed up.

On the other side, There were 225 students who our model said would not show up, who actually attended. And last, there were 1251 students who our model said would attend who actually did– we correctly classified 85 percent of actual attendees. The overall accuracy of our model ends up being (312+1251)/2150 or 72 percent.

Question: is this a good model?

Prediction via Linear Regression: Wrong, but Useful!

We can use our standard tool of linear regression to build an model and make predictions, with just a few adjustments. This will be wrong, but useful.

We'll use the wrong model, a linear regression. Running a linear regression with a binary dependent variable is called a linear probability model, which ironically it is not.

We'll use a formula that includes the variables we've used so far.

```
admit_formula<-as.formula("yield~sat+net_price+legacy")

ad_analysis <- ad %>%
  ungroup() %>%
  select(yield,sat,net_price,legacy) %>%
  drop_na()

m <- lm(formula = admit_formula,data = ad_analysis)
summary(m)
```

```
##
## Call:
## lm(formula = admit_formula, data = ad_analysis)
##
## 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
```

Question: What do we make of these coefficients

To evaluate our model, we traditionally have relied on RMSE. However, this is less appropriate when using a binary dependent variable. Instead, we want to think about **accuracy**. Let's start by getting the predictions, as always.

```
ad_analysis <- ad_analysis %>%
  mutate(preds = predict(m)) %>%
  mutate(errors = yield - preds)
```

Let's take a look at these predictions

```
ad_analysis %>% select(yield,preds)
```

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

So these are probabilities. To complete the classification problem, we need to assign group labels to each case in the testing dataset. Let's assume that a probability equal to or greater than .5 will be classified as a 1 and everything else as a 0.

```
ad_analysis<-ad_analysis%>%
  mutate(pred_attend=ifelse(preds>=.5,1,0))

ad_analysis%>%select(yield,preds,pred_attend)
```

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

Here's the problem with using a linear regression in this case: there's no guarantee that the results will be on the probability scale. So, we can find cases where our model predicted probabilities below 0 or above 1. Of course, these just get labeled as 1 or 0.

```
ad_analysis%>%
  filter(preds>1|preds<0)%>%
  select(yield,preds,pred_attend)
```

```
## # A tibble: 128 × 3
##    yield preds pred_attend
##    <int> <dbl>       <dbl>
##  1     1  1.07           1
##  2     1  1.06           1
##  3     1  1.02           1
##  4     1  1.10           1
##  5     1  1.03           1
##  6     1  1.12           1
##  7     1  1.18           1
##  8     1  1.49           1
##  9     1  1.15           1
## 10     1  1.11           1
## # ℹ 118 more rows
```

Accuracy of Linear Regression

```
ad_analysis%>%
  group_by(yield)%>%
  mutate(total_attend=n())%>%
  group_by(yield,pred_attend)%>%
  summarize(n(),`Actual Group`=mean(total_attend))%>%
  mutate(Proportion=`n()`/`Actual Group`)%>%
  rename(`Actually Attended`=yield,
         `Predicted to Attend`=pred_attend,
         `Number of Students`=`n()`)
```

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

```
## # A tibble: 4 × 5
## # Groups:   Actually Attended [2]
##   `Actually Attended` `Predicted to Attend` `Number of Students` `Actual Group`
##                 <int>                 <dbl>                <int>          <dbl>
## 1                   0                     0                  282            684
## 2                   0                     1                  402            684
## 3                   1                     0                  113           1466
## 4                   1                     1                 1353           1466
## # ℹ 1 more variable: Proportion <dbl>
```

In this model, we correctly classified 282 out of 684 non-attendees or about 41 percent, and 1353 out of 1466 attendees or about 92 percent. The overall accuracy is (282+1353)/2150 or 76 percent. How are we doing?

Sensitivity

In the above table, the percent of 1s correctly identified is a measure known as **sensitivity**:

```
ad_analysis%>%
  group_by(yield)%>%
  mutate(total_attend=n())%>%
  group_by(yield,pred_attend)%>%
  summarize(n(),`Actual Group`=mean(total_attend))%>%
  mutate(Proportion=`n()`/`Actual Group`)%>%
  rename(`Actually Attended`=yield,
         `Predicted to Attend`=pred_attend,
         `Number of Students`=`n()`) %>%
  filter(`Actually Attended` == 1 & `Predicted to Attend` == 1)
```

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

```
## # A tibble: 1 × 5
## # Groups:   Actually Attended [1]
##   `Actually Attended` `Predicted to Attend` `Number of Students` `Actual Group`
##                 <int>                 <dbl>                 <int>          <dbl>
## 1                   1                     1                  1353           1466
## # ℹ 1 more variable: Proportion <dbl>
```

Specificity

The percent of 0s correctly identified is a measure know as **specificity**

```
ad_analysis%>%
  group_by(yield)%>%
  mutate(total_attend=n())%>%
  group_by(yield,pred_attend)%>%
  summarize(n(),`Actual Group`=mean(total_attend))%>%
  mutate(Proportion=`n()`/`Actual Group`)%>%
  rename(`Actually Attended`=yield,
         `Predicted to Attend`=pred_attend,
         `Number of Students`=`n()`) %>%
  filter(`Actually Attended` == 0 & `Predicted to Attend` == 0)
```

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

```
## # A tibble: 1 × 5
## # Groups:   Actually Attended [1]
##   `Actually Attended` `Predicted to Attend` `Number of Students` `Actual Group`
##                 <int>                 <dbl>                 <int>          <dbl>
## 1                   0                     0                   282            684
## # ℹ 1 more variable: Proportion <dbl>
```

Accuracy

```
ad_analysis%>%
  group_by(yield)%>%
  mutate(total_attend=n())%>%
  group_by(yield,pred_attend)%>%
  summarize(n(),`Actual Group`=mean(total_attend))%>%
  mutate(Proportion=`n()`/`Actual Group`)%>%
  rename(`Actually Attended`=yield,
         `Predicted to Attend`=pred_attend,
         `Number of Students`=`n()`) %>%
  filter(`Actually Attended` == `Predicted to Attend`) %>%
  ungroup() %>%
  summarise(Accuracy = sum(`Number of Students`) / sum(`Actual Group`))
```

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

```
## # A tibble: 1 × 1
##   Accuracy
##      <dbl>
## 1    0.760
```

Logistic Regression

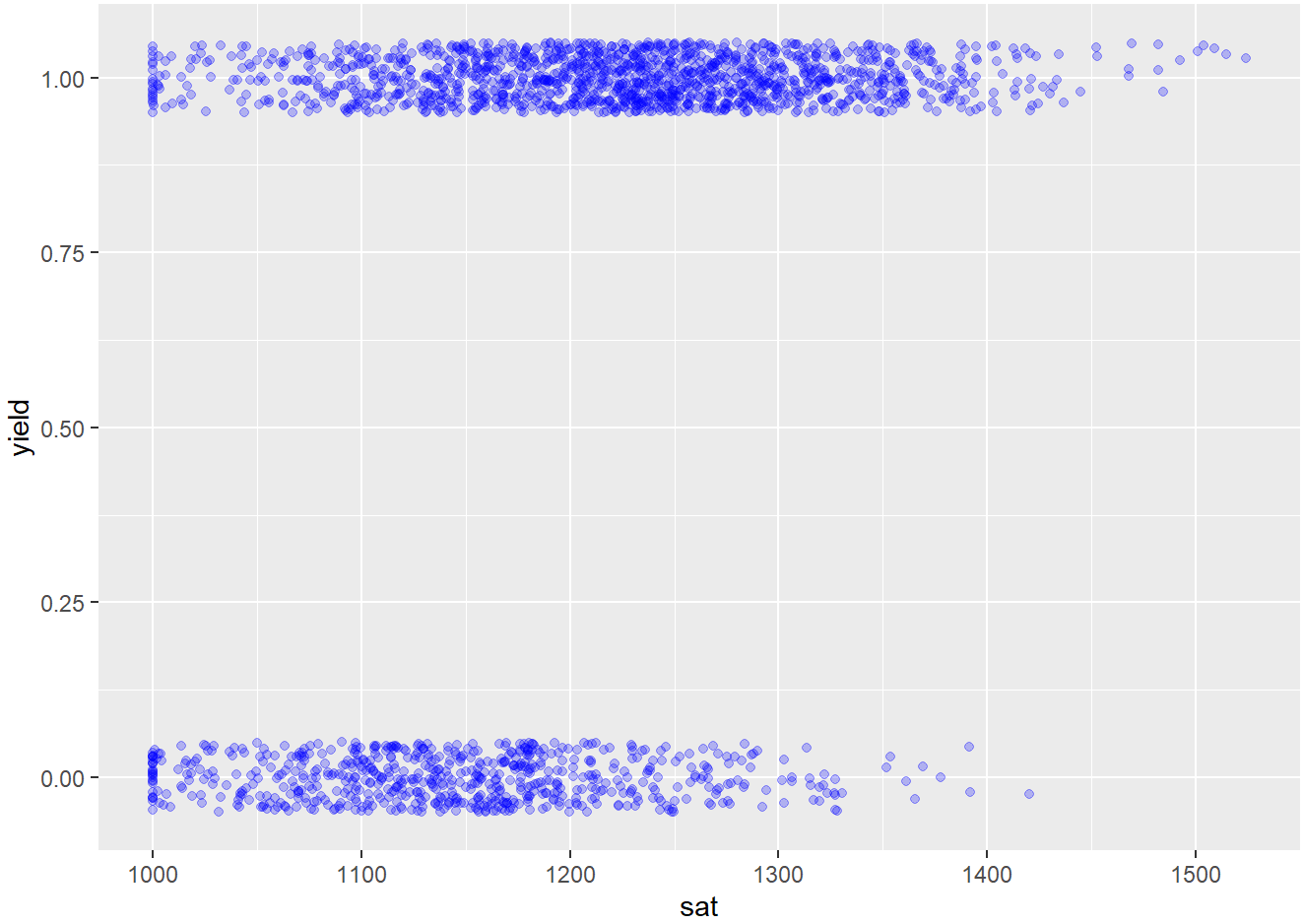
So far, we've been using tools we know for classification. While we *can* use conditional means or linear regression for classification, it's better to use a tool that was created specifically for binary outcomes.

Logistic regression is set up to handle binary outcomes as the dependent variable. The downside to logistic regression is that it is modeling the log odds of the outcome, which means all of the coefficients are expressed as log odds, which (almost) no one understands intuitively.

Let's take a look at a simple plot of our dependent variable as a function of one independent variable: SAT scores. I'm using `geom_jitter` to allow the points to "bounce" around a bit on the y axis so we can actually see them, but it's important to note that they can only be 0s or 1s.

Yield as a Function of SAT Scores

```
ad%>%
  ggplot(aes(x=sat,y=yield))+
  geom_jitter(width=.01,height=.05,alpha=.25,color="blue")
```

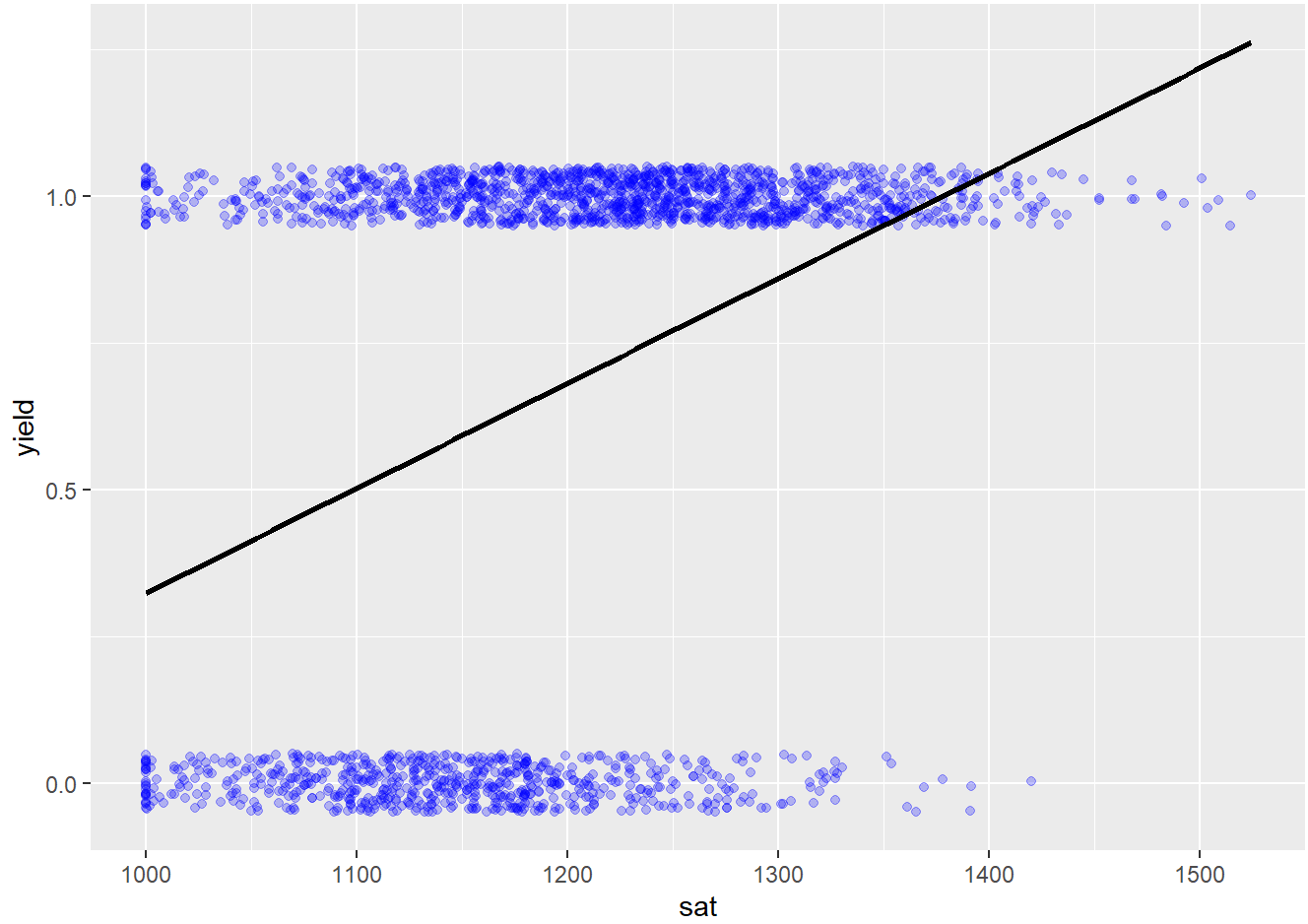


We can see there are more higher SAT students than lower SAT students that ended up enrolling. A linear model in this case would look like this:

Predicted “Probabilities” from a Linear Model

```
ad%>%
  ggplot(aes(x=sat,y=yield))+
  geom_jitter(width=.01,height=.05,alpha=.25,color="blue")+
  geom_smooth(method="lm",se = FALSE,color="black")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



We can see the issue we identified last time: we CAN fit a model, but it doesn't make a ton of sense. In particular, it doesn't follow the data very well and it ends up with probabilities outside 0,1.

Generalized Linear Models

What we need is a better function that connects y to x. The idea of connecting y to x with a function other than a simple line is called a generalized linear model.

A posits that the probability that y is equal to some value is a function of the independent variables and the coefficients or other parameters via a *link function*:

$P(y|\mathbf{x}) = G(\beta_0 + \mathbf{x_i}\beta)$

In our case, we're interested in the probability that $y = 1$

$P(y = 1|\mathbf{x}) = G(\beta_0 + \mathbf{x_i}\beta)$

There are several functions that "map" onto a 0,1 continuum. The most commonly used is the logistic function, which gives us the *logit model*.

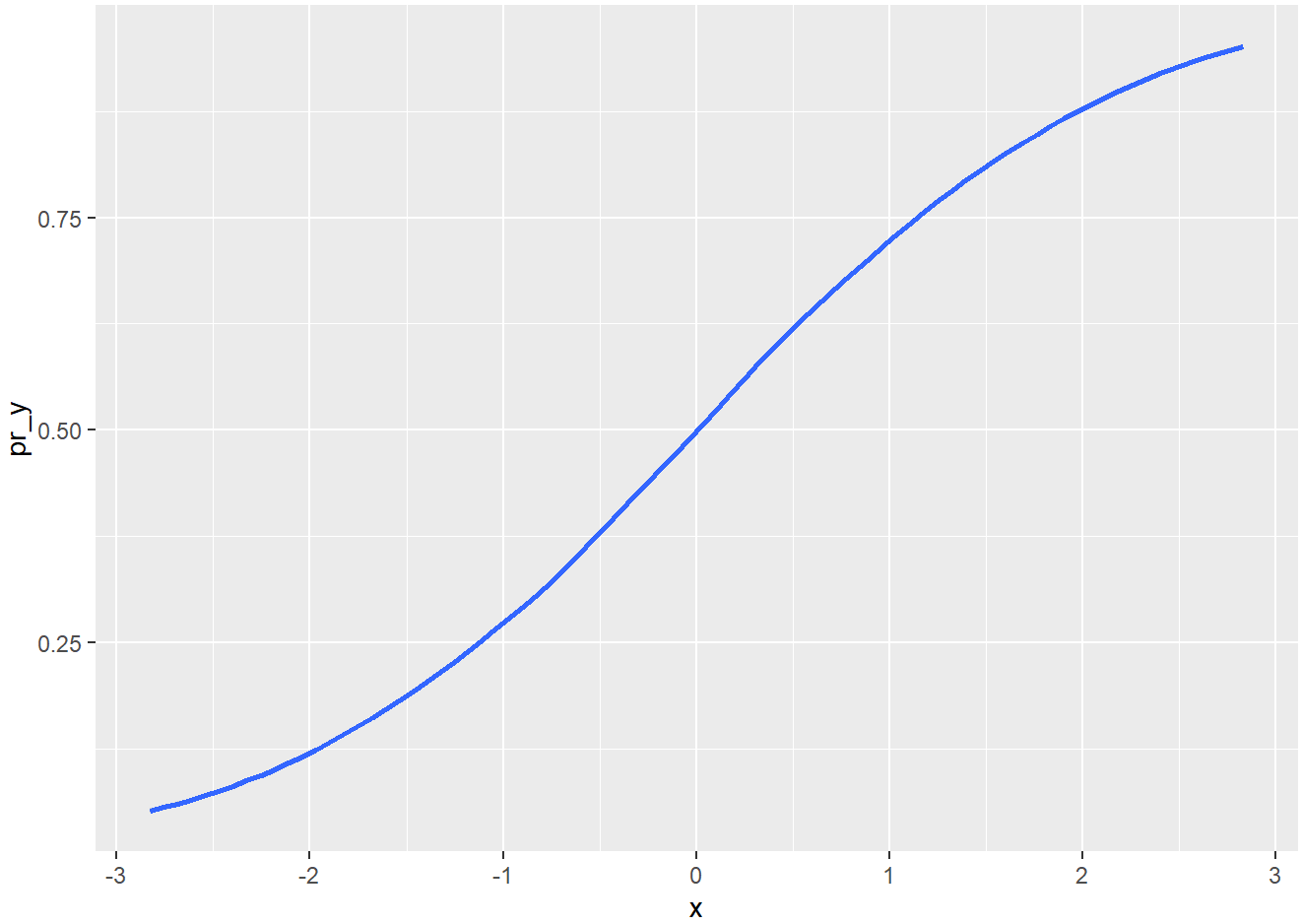
The logistic function is given by:

$f(x) = \frac{1}{1+exp^{-k(x-x_0)}}$

The Logistic Function: Pr(Y) as a Function of X

```
x<-runif(100,-3,3)
pr_y=1/(1+exp(-x))
as_tibble(pr_y = pr_y,x = x)%>%
  ggplot(aes(x=x,y=pr_y))+
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Mapped onto our GLM, this gives us:

$$P(y = 1|\mathbf{x}) = \frac{exp(\beta_0 + \mathbf{x_i}\beta)}{1 + exp(\beta_0 + \mathbf{x_i}\beta)}$$

The critical thing to note about the above is that the link function maps the entire result of estimation $(\beta_0 + \mathbf{x_i}\beta)$ onto the 0,1 continuum. Thus, the change in the $P(y = 1|\mathbf{x})$ is a function of *all* of the independent variables and coefficients together, one at a time.

What does this mean? It means that the coefficients can only be interpreted on the *logit* scale, and don't have the normal interpretation we would use for OLS regression. Instead, to understand what the logistic regression coefficients mean, you're going to have to convert the entire term $(\beta_0 + \mathbf{x_i}\beta)$ to the probability scale, using the inverse of the function. Luckily we have computers to do this for us . . .

If we use this link function on our data, it would look like this: `glm(formula,family,data)`. Notice that it looks very similar to the linear regression function: `lm(formula,data)`. The only difference is the **name** of the function (`glm()` versus `lm()`) and the additional input `family`. This input can take on many different values, but for this class, we only want the **logit**, which requires `family = binomial(link = "logit")`.

Putting it all together:

Plotting Predictions from Logistic Regression

```
m <- glm(yield ~ sat, family = binomial(link = "logit"), data = ad_analysis)# %>% ## Run a glm
summary(m)
```

```
##
## Call:
## glm(formula = yield ~ sat, family = binomial(link = "logit"),
##     data = ad_analysis)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.077e+01  6.965e-01  -15.46   <2e-16 ***
## sat          9.730e-03  5.926e-04   16.42   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2689.5  on 2149  degrees of freedom
## Residual deviance: 2353.7  on 2148  degrees of freedom
## AIC: 2357.7
##
## Number of Fisher Scoring iterations: 4
```

How to interpret? It is more complicated than a linear regression model, and beyond what you are expected to know in an intro to data science class. We **cannot** say that each additional SAT score point corresponds to a 0.000973 increase in `yield`. However, we can conclude that there is a (1) positive and (2) statistically significant association between SAT scores and attending.

In this class…just focus on the **sign** of the coefficient and the **p-value**.

Quick Exercise: Replicate the above model using distance as a predictor and comment on what it tells you

```
## INSERT CODE HERE
```

As you're getting started, this is what we recommend with these models:

- Use coefficient estimates for sign and significance only–don't try and come up with a substantive interpretation.
- Generate probability estimates based on characteristics for substantive interpretations.

Evaluating

Since the outcome is binary, we want to evaluate our model on the basis of **sensitivity**, **specificity**, and **accuracy**. To get started, let's generate predictions again.

NOTE: when predicting a `glm()` model, set `type = "response"` !

```
m <- glm(yield ~ sat, family = binomial(link = "logit"), data = ad_analysis)# %>% ## Run a glm
summary(m)
```

```
##
## Call:
## glm(formula = yield ~ sat, family = binomial(link = "logit"),
##     data = ad_analysis)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.077e+01  6.965e-01  -15.46   <2e-16 ***
## sat          9.730e-03  5.926e-04   16.42   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2689.5  on 2149  degrees of freedom
## Residual deviance: 2353.7  on 2148  degrees of freedom
## AIC: 2357.7
##
## Number of Fisher Scoring iterations: 4
```

```
ad_analysis%>%
  mutate(preds = predict(m,type = 'response')) %>% # Predicting our new model
  mutate(pred_attend = ifelse(preds > .5,1,0)) %>% # Converting predicted probabilities into 1s
and 0s
  group_by(yield)%>%
  mutate(total_attend=n())%>%
  group_by(yield,pred_attend)%>%
  summarize(n(),`Actual Group`=mean(total_attend))%>%
  mutate(Proportion=`n()`/`Actual Group`)%>%
  rename(`Actually Attended`=yield,
         `Predicted to Attend`=pred_attend,
         `Number of Students`=`n()`)
```

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

```
## # A tibble: 4 × 5
## # Groups:   Actually Attended [2]
##   `Actually Attended` `Predicted to Attend` `Number of Students` `Actual Group`
##                 <int>                 <dbl>                 <int>          <dbl>
## 1                   0                     0                   220            684
## 2                   0                     1                   464            684
## 3                   1                     0                   173           1466
## 4                   1                     1                  1293           1466
## # ℹ 1 more variable: Proportion <dbl>
```

Our **sensitivity** is 0.88 or 88%, our **specificity** is 0.32 or 32%, and our overall **accuracy** is `(220 + 1293) / 2150` or 0.70 (70%).

The Thresholds

Note that we required a “threshold” to come up with these measures of sensitivity, specificity, and accuracy. Specifically, we relied on a coin toss. If the predicted probability of attending was greater than 50%, we assumed that the student would attend, otherwise they wouldn’t. But this choice doesn’t always have to be 50%. We can choose a number of different thresholds.

```
ad_analysis%>%
  mutate(preds = predict(m,type = 'response')) %>% # Predicting our new model
  mutate(pred_attend = ifelse(preds > .35,1,0)) %>% # A lower threshold of 0.35 means that more
students are predicted to attend
  group_by(yield)%>%
  mutate(total_attend=n())%>%
  group_by(yield,pred_attend)%>%
  summarize(n(),`Actual Group`=mean(total_attend))%>%
  mutate(Proportion=`n()`/`Actual Group`)%>%
  rename(`Actually Attended`=yield,
         `Predicted to Attend`=pred_attend,
         `Number of Students`=`n()`)
```

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

```
## # A tibble: 4 × 5
## # Groups:   Actually Attended [2]
##   `Actually Attended` `Predicted to Attend` `Number of Students` `Actual Group`
##                 <int>                 <dbl>                 <int>          <dbl>
## 1                   0                     0                    73            684
## 2                   0                     1                   611            684
## 3                   1                     0                    51           1466
## 4                   1                     1                  1415           1466
## # ℹ 1 more variable: Proportion <dbl>
```

```
ad_analysis%>%
  mutate(preds = predict(m,type = 'response')) %>% # Predicting our new model
  mutate(pred_attend = ifelse(preds > .75,1,0)) %>% # A higher threshold of 0.75 means that fewe
r students are predicted to attend
  group_by(yield)%>%
  mutate(total_attend=n())%>%
  group_by(yield,pred_attend)%>%
  summarize(n(),`Actual Group`=mean(total_attend))%>%
  mutate(Proportion=`n()`/`Actual Group`)%>%
  rename(`Actually Attended`=yield,
         `Predicted to Attend`=pred_attend,
         `Number of Students`=`n()`)
```

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

```
## # A tibble: 4 × 5
## # Groups:   Actually Attended [2]
##   `Actually Attended` `Predicted to Attend` `Number of Students` `Actual Group`
##                 <int>                 <dbl>                <int>          <dbl>
## 1                   0                     0                  566            684
## 2                   0                     1                  118            684
## 3                   1                     0                  661           1466
## 4                   1                     1                  805           1466
## # ℹ 1 more variable: Proportion <dbl>
```

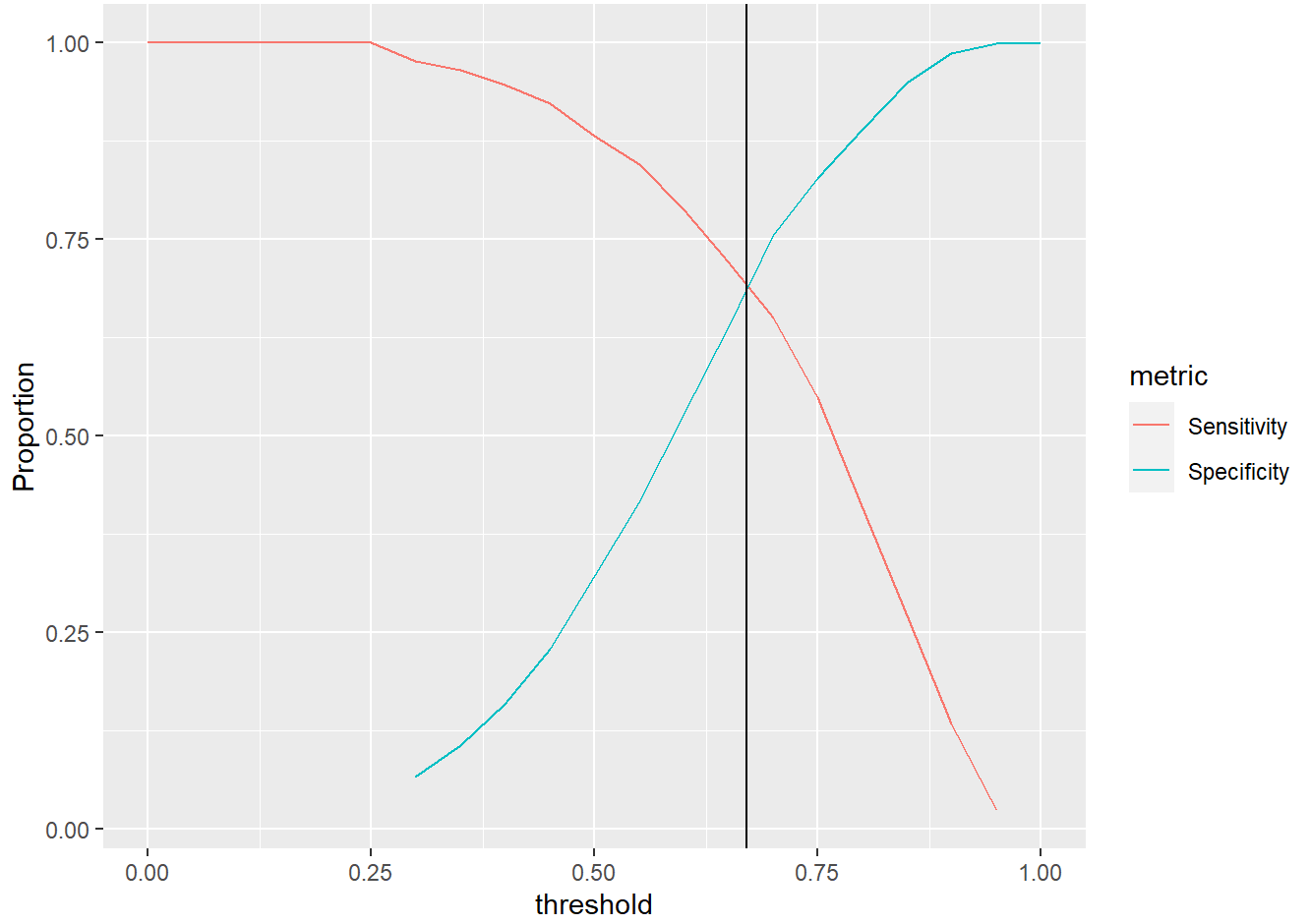
So how do we determine the optimal threshold? Loop over different choices!

```
threshRes <- NULL

for(thresh in seq(0,1,by = .05)) { # Loop over values between zero and one, increasing by 0.05
  tmp <- ad_analysis%>%
  mutate(preds = predict(m,type = 'response')) %>% # Predicting our new model
  mutate(pred_attend = ifelse(preds > thresh,1,0)) %>% # Plug in our threshold value
  group_by(yield)%>%
  mutate(total_attend=n())%>%
  group_by(yield,pred_attend)%>%
  summarize(n(),`Actual Group`=mean(total_attend),.groups = 'drop')%>%
  mutate(Proportion=`n()`/`Actual Group`)%>%
  rename(`Actually Attended`=yield,
         `Predicted to Attend`=pred_attend,
         `Number of Students`=`n()`) %>%
    mutate(threshold = thresh)

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

threshRes %>%
  mutate(metric = ifelse(`Actually Attended` == 1 & `Predicted to Attend` == 1,'Sensitivity',
                         ifelse(`Actually Attended` == 0 & `Predicted to Attend` == 0,'Specifici
ty',NA))) %>%
  drop_na() %>%
  ggplot(aes(x = threshold,y = Proportion,color = metric)) +
  geom_line() +
  geom_vline(xintercept = .67)
```



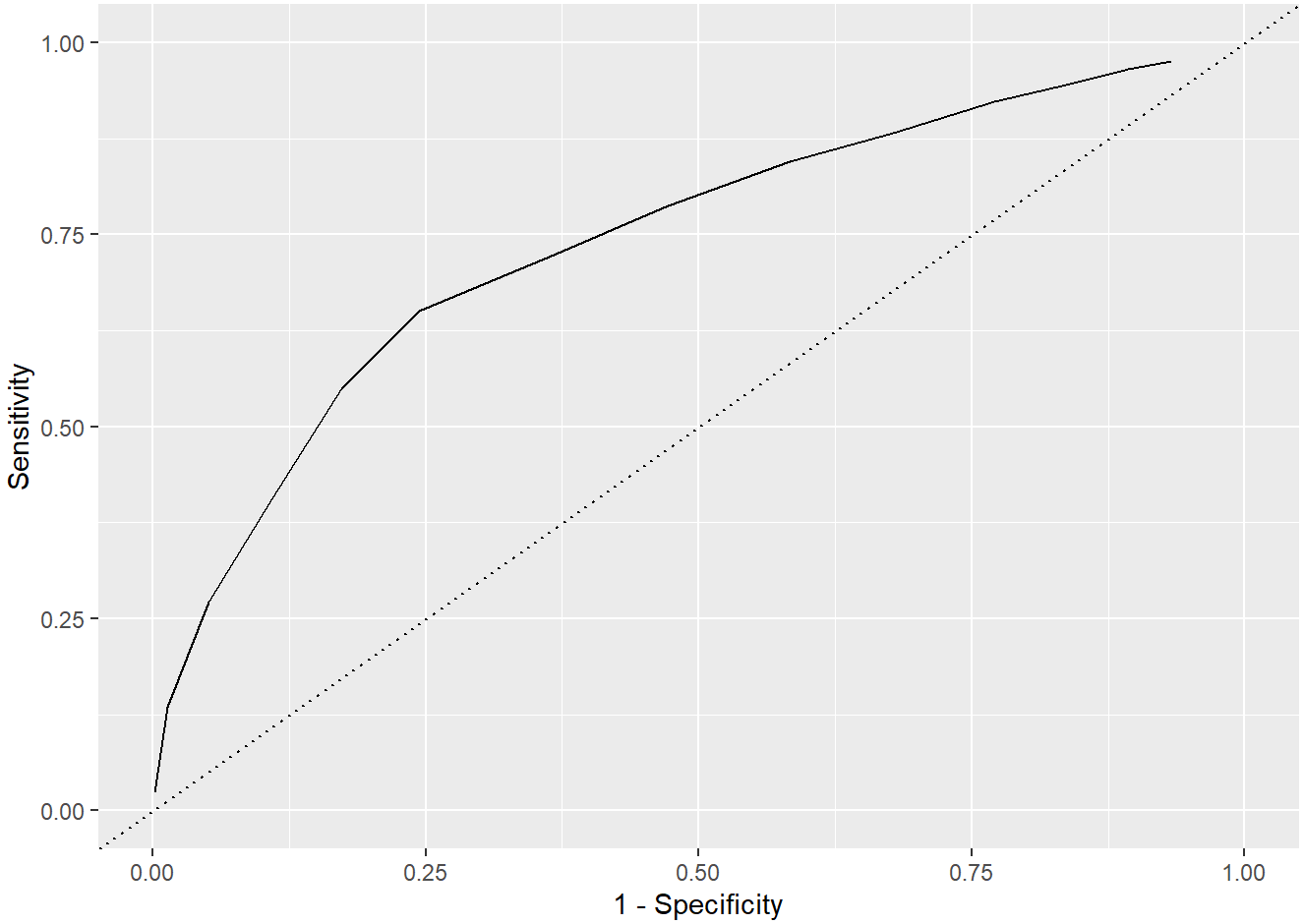
The optimal threshold is the one that maximizes Sensitivity and Specificity! (Although this is use-case dependent. You might prefer to do better on accurately predicting those who do attend than you do about predicting those who don't.) In this case it is about 0.67.

The ROC curve

As the preceding plot makes clear, there is a **trade-off** between sensitivity and specificity. We can visualize this trade-off by putting `1-specificity` on the x-axis, and `sensitivity` on the y-axis, to create the "Receiver-Operator Characteristic (ROC) Curve".

```
threshRes %>%
  mutate(metric = ifelse(`Actually Attended` == 1 & `Predicted to Attend` == 1,'Sensitivity',
                         ifelse(`Actually Attended` == 0 & `Predicted to Attend` == 0,'Specifici
ty',NA))) %>%
  drop_na() %>%
  select(Proportion,metric,threshold) %>%
  spread(metric,Proportion) %>% # Create two columns, one for spec, the other for sens
  ggplot(aes(x = 1-Specificity,y = Sensitivity)) + # X-axis is 1-Specificity
  geom_line() +
  xlim(c(0,1)) + ylim(c(0,1)) +
  geom_abline(slope = 1,intercept = 0,linetype = 'dotted') # The curve is always evaluated in re
ference to the diagonal line.
```

```
## Warning: Removed 7 rows containing missing values (`geom_line()`).
```



The idea is that a model that is very predictive will have high levels of sensitivity AND high levels of specificity at EVERY threshold. Such a model will cover most of the available area above the baseline of .5. A model with low levels of sensitivity and low levels of specificity at every threshold will cover almost none of the available area above the baseline of .5.

As such, we can extract a single number that captures the quality of our model from this plot: the “Area Under the Curve” (AUC). The further away from the diagonal line is our ROC curve, the better our model performs, and the higher is the AUC. But how to calculate the AUC? Thankfully, there is a helpful `R` package that will do this for us: `tidymodels`.

```
require(tidymodels)
```

```
## Loading required package: tidymodels
```

```
## ── Attaching packages ─────────────────────────────────────── tidymodels 1.1.1 ──
```

```
## ✓ broom        1.0.5     ✓ rsample      1.2.0
## ✓ dials        1.2.0     ✓ tune         1.1.2
## ✓ infer        1.0.5     ✓ workflows    1.1.3
## ✓ modeldata    1.2.0     ✓ workflowsets 1.0.1
## ✓ parsnip      1.1.1     ✓ yardstick    1.2.0
## ✓ recipes      1.0.8
```

```
## ── 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 suppressPackageStartupMessages() to eliminate package startup messages
```

```
roc_auc(data = ad_analysis %>%
  mutate(pred_attend = predict(m,type = 'response'),
         truth = factor(yield,levels = c('1','0'))) %>% # Make sure the outcome is converted to
factors with '1' first and '0' second!
  select(truth,pred_attend),truth,pred_attend)
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.742
```

Our curve covers almost 75% of the total area above the diagonal line. Is this good?

Just like with RMSE, you are primarily interested in the AUC measure to compare different models against each other.

But if you HAVE to, it turns out that – in general – interpreting AUC is just like interpreting academic grades:

Below .6= bad (F)

.6-.7= still not great (D)

.7-.8= Ok . .. (C)

.8-.9= Pretty good (B)

.9-1= Very good fit (A)

Quick Exercise: Rerun the model with sent_scores added: does it improve model fit?

Cross Validation

Just like RMSE calculated on the full data risks overfitting, AUC does also.

How to overcome? Cross validation!

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

  # Training models
  m1 <- glm(yield ~ sat,family = binomial(link = "logit"),train)

  # Predicting models
  toEval <- test %>%
    mutate(m1Preds = predict(m1,newdata = test,type = 'response'),
           truth = factor(yield,levels = c('1','0')))

  # Evaluating models
  rocRes <- roc_auc(data = toEval,truth = truth,m1Preds)
  cvRes <- rocRes %>%
    mutate(bsInd = i) %>%
    bind_rows(cvRes)
}

mean(cvRes$.estimate)
```

```
## [1] 0.7404506
```