Review Session 10

Sophie Hill

04/08/2022

PSet 3

Last one!

Q1-Q10: Difference-in-differences (RS9)

Protip: Read the intro and skim the Card & Krueger paper!

Q11-Q14:

- Interpreting coefficients (RS7)
- Omitted variable bias (RS8)
- Fixed effects (RS9)

Q15-Q17: Predicting a numeric outcome (RS10)

Q18-Q20: Predicting a binary outcome, logistic regression (RS10)

Agenda

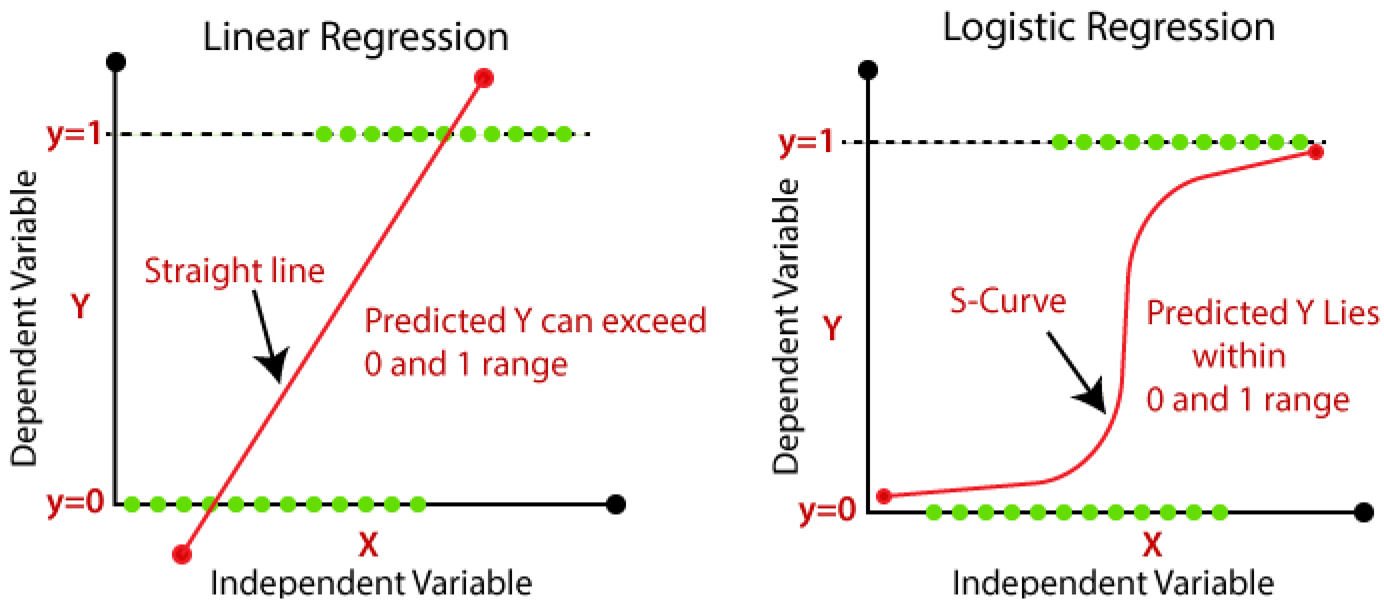
Logistic regression

- Why?
- Interpreting coefficients
- Predicted probabilities

Prediction

- Classification threshold
- Accuracy, precision, recall
- For numeric outcomes: RMSE, MAE, MAPE

Logistic regression: Why?



Logistic regression: R code

```
# Logistic regression
glm(vote_dem ~ age + white,
    data = cces20,
    family = "binomial") %>%
  tidy()
```

```
## # A tibble: 3 × 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)   2.07    0.0394        52.5 0
## 2 age          -0.0189  0.000629     -30.1 2.10e-198
## 3 white        -0.858   0.0250       -34.3 9.43e-258
```

How is this different to linear regression with `lm()`?

- `glm()` instead of `lm()`
- `family = "binomial"` tells R that the outcome variable is binary
- interpretation of the coefficients is different

Interpreting logistic coefficients

Recall that the logit function is defined by:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

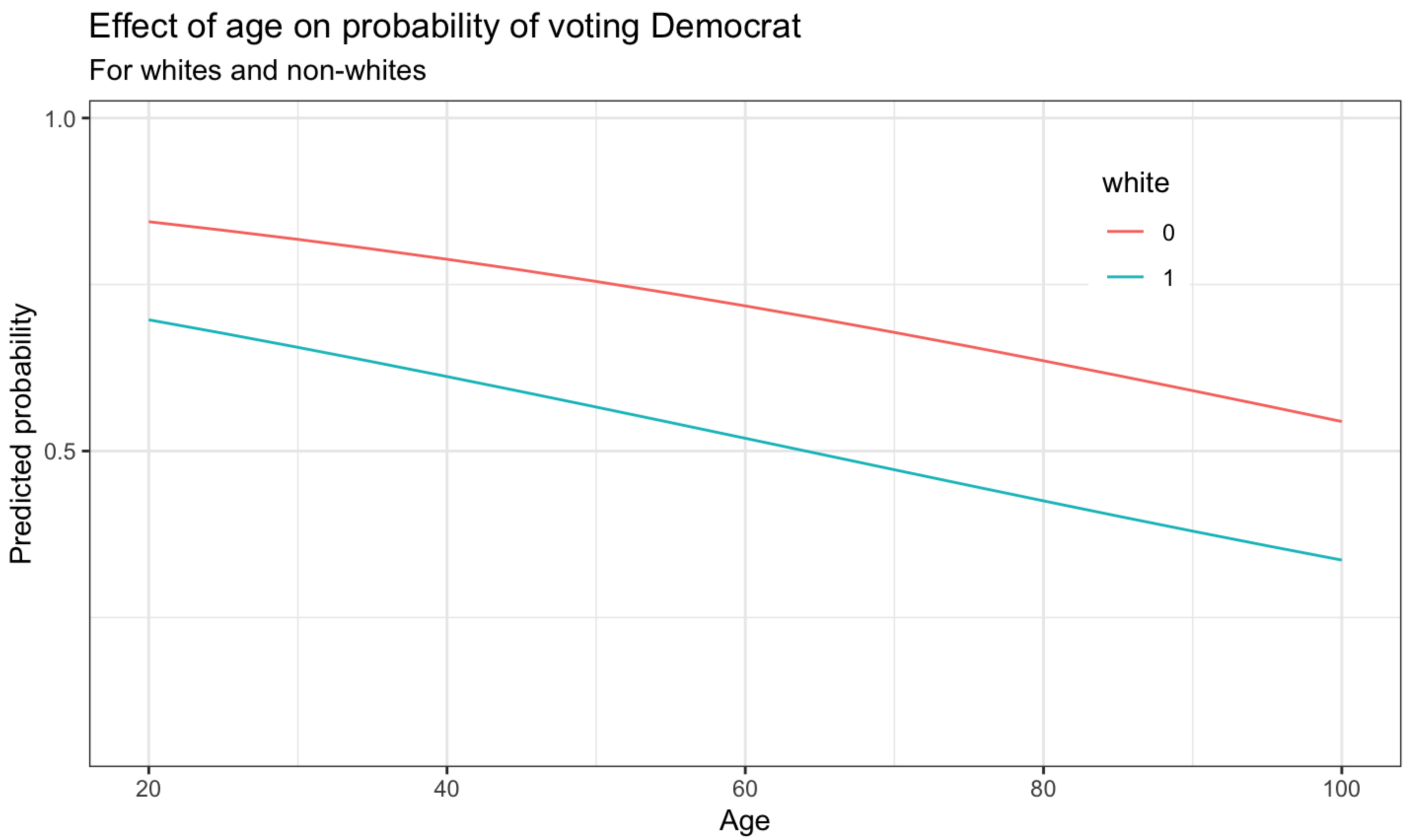
So this is our regression equation:

$$\text{logit}\left[\widehat{P(\text{vote\_dem} = 1)}\right] = 2.069 - 0.019(\text{age}) - 0.858(\text{white})$$

Or, equivalently:

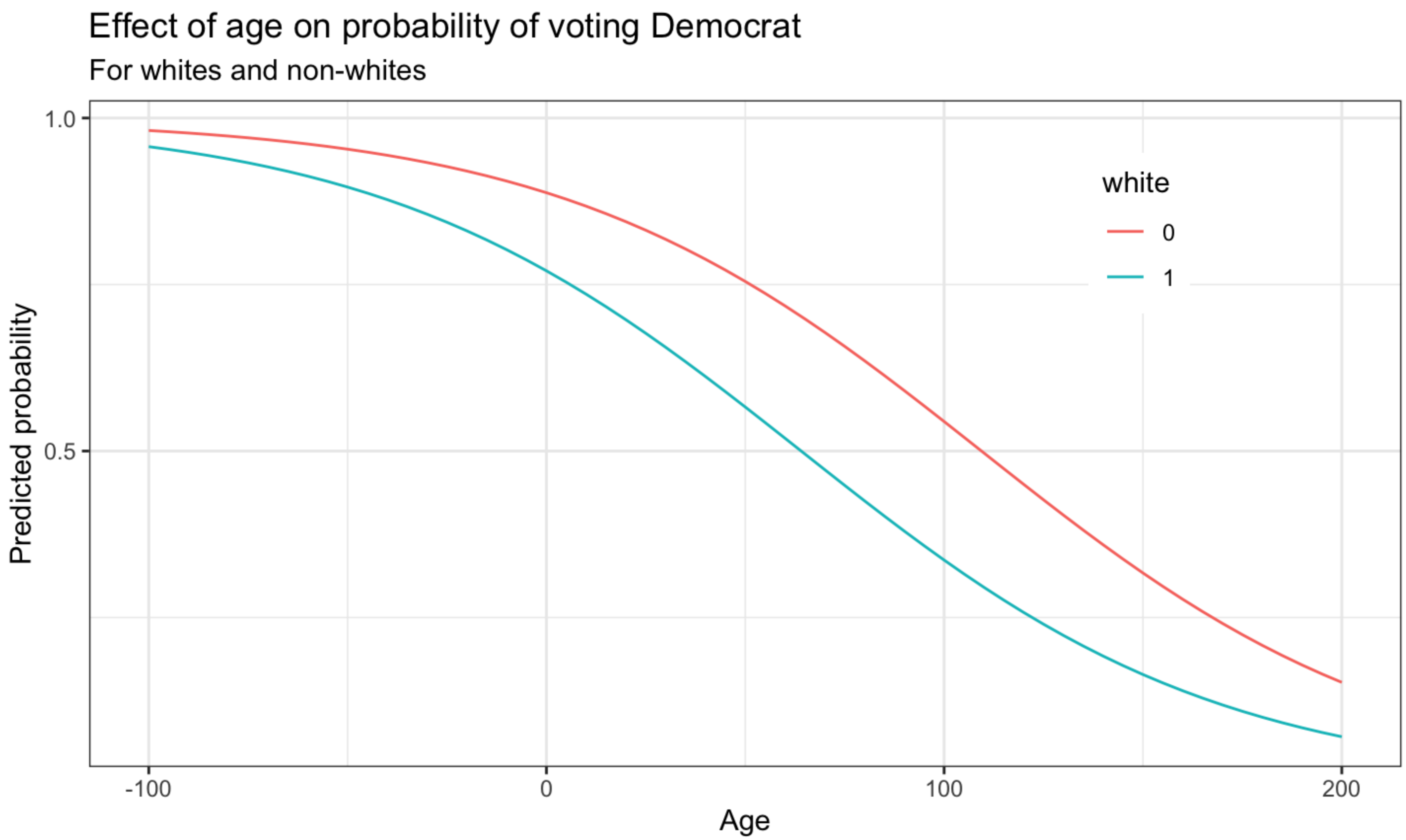
$$\widehat{P(\text{vote\_dem} = 1)} = \text{logit}^{-1}\left[2.069 - 0.019(\text{age}) - 0.858(\text{white})\right]$$

The S-curve



Effect of age on probability of voting Democrat
For whites and non-whites
white
0
1
1.0
0.5
Predicted probability
20
40
60
80
100
Age

The S-curve


Effect of age on probability of voting Democrat
For whites and non-whites
1.0
0.5
Predicted probability
-100
0
100
200
Age
white
0
1

The “divide-by-4” trick

$$\text{logit}\left[\widehat{P(\text{vote\_dem} = 1)}\right] = 2.069 - 0.019(\text{age}) - 0.858(\text{white})$$

Useful trick: Divide a logit coefficient by 4 to get an upper bound of the ppt change in $P(Y_i = 1)$.

$-0.019/4 = -0.005 \qquad\qquad -0.858/4 = -0.215$

- Increasing **age** by 1 year is associated with no more than a 0.5ppt decrease in the probability of voting democratic.
- Being **white** (vs non-white) is associated with no more than a 22ppt decrease in the probability of voting democratic.

Caution: this is just a rough rule-of-thumb. It is more accurate when the outcome probability is close to 0.5 (where the S-curve is the steepest).

Change in predicted probabilities

$$\text{logit}\left[\widehat{P(\text{vote\_dem} = 1)}\right] = 2.069 - 0.019(\text{age}) - 0.858(\text{white})$$

The divide-by-4 rule gives us a rough upper bound. To interpret the coefficient on X_i more precisely, we should compute predicted probabilities at different values of X_i.

The effect of a 1-unit change in X_i can vary depending on the values of the other predictors. (This is a non-linear model!) So we also need to pick some sensible values to hold the other predictors constant at.

Change in predicted probabilities

```
ggpredict(mod_logistic,
          terms = c("age [20:80, by = 10]",
                    "white [1]"))
```

```
## # Predicted probabilities of vote_dem
##
## age | Predicted |       95% CI
## ------------------------------
##  20 |      0.70 | [0.69, 0.71]
##  30 |      0.66 | [0.65, 0.66]
##  40 |      0.61 | [0.61, 0.62]
##  50 |      0.57 | [0.56, 0.57]
##  60 |      0.52 | [0.51, 0.52]
##  70 |      0.47 | [0.46, 0.48]
##  80 |      0.43 | [0.42, 0.43]
```

Change in predicted probabilities

```
ggpredict(mod_logistic,
          terms = c("white"))
```

```
## # Predicted probabilities of vote_dem
##
## white | Predicted |       95% CI
## --------------------------------
##     0 |      0.74 | [0.74, 0.75]
##     1 |      0.55 | [0.55, 0.56]
##
## Adjusted for:
## * age = 53.14
```

Classification

```
augment(mod_logistic,
        cces20,
        type.predict = "response") %>%
  select(1:4) %>%
  head()
```

```
## # A tibble: 6 × 4
##     age white vote_dem .fitted
##   <dbl> <dbl>    <dbl>   <dbl>
## 1    54     1        0   0.547
## 2    74     1        1   0.453
## 3    58     1        1   0.529
## 4    59     1        0   0.524
## 5    73     1        0   0.458
## 6    50     1        0   0.566
```

Choosing the threshold

```r
cces20 <-
  augment(mod_logistic, cces20,
          type.predict = "response") %>%
  select(1:4) %>%
  mutate(pred = ifelse(.fitted > 0.5, 1, 0))

head(cces20)
```

```
## # A tibble: 6 × 5
##     age white vote_dem .fitted  pred
##   <dbl> <dbl>    <dbl>   <dbl> <dbl>
## 1    54     1        0   0.547     1
## 2    74     1        1   0.453     0
## 3    58     1        1   0.529     1
## 4    59     1        0   0.524     1
## 5    73     1        0   0.458     0
## 6    50     1        0   0.566     1
```

How good are our predictions?

Accuracy: out of all predictions, what % are correct?

$$\frac{TP+TN}{TP+FP+TN+FN}$$

Precision: out of all *predicted* 1's, what % are correct?

$$\frac{TP}{TP+FP}$$

Recall: out of all *actual* 1's, what % are correctly predicted?

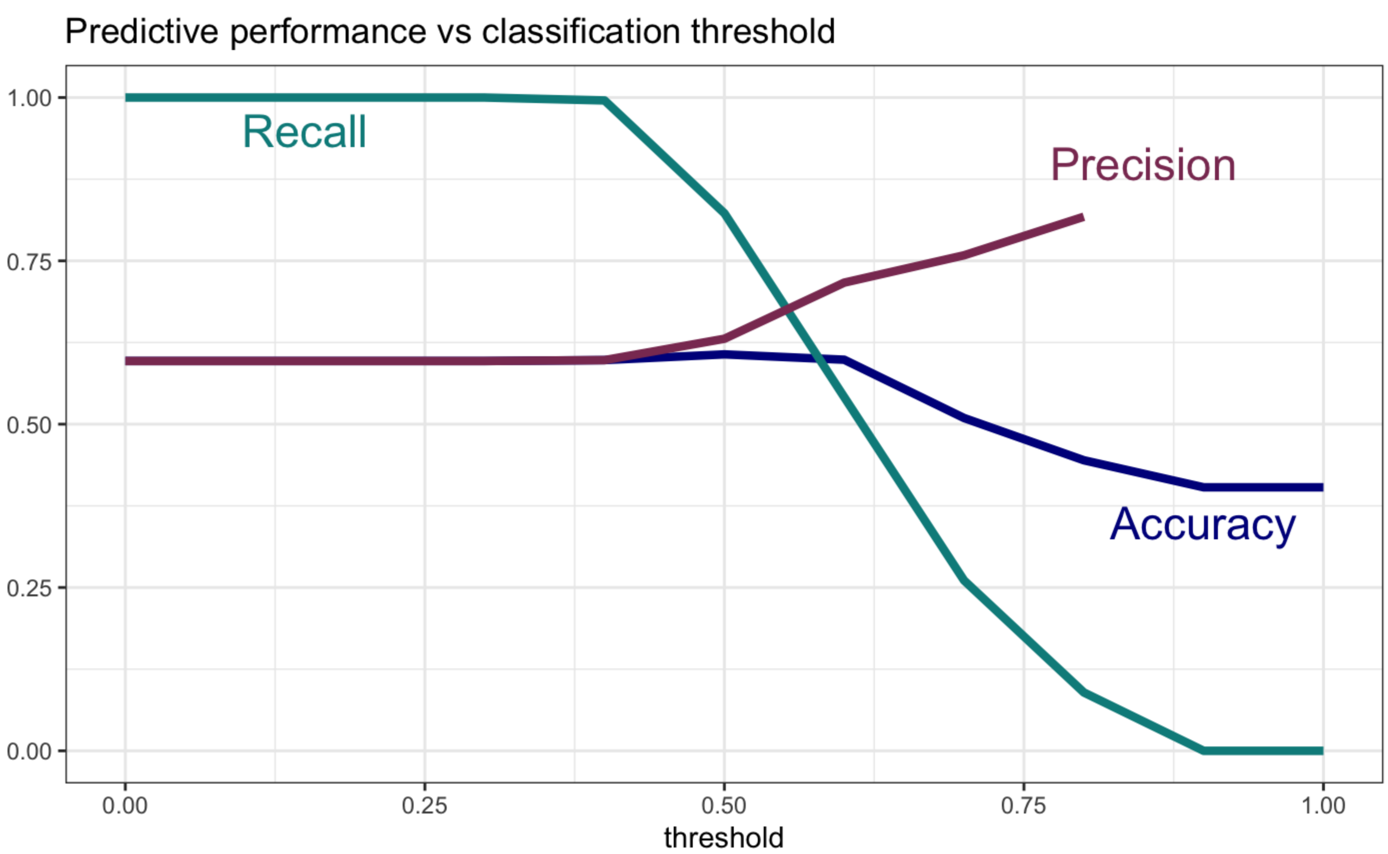
$$\frac{TP}{TP+FN}$$

How good are our predictions?

```
cces20 %>%
  summarize(
    accuracy = sum(vote_dem == pred) / nrow(cces20),
    precision = sum(vote_dem == 1 & pred == 1) / sum(pred == 1),
    recall = sum(vote_dem == 1 & pred == 1) / sum(vote_dem == 1)
  )
```

```
## # A tibble: 1 × 3
##   accuracy precision recall
##      <dbl>     <dbl>  <dbl>
## 1    0.607     0.631  0.823
```

How good are our predictions?



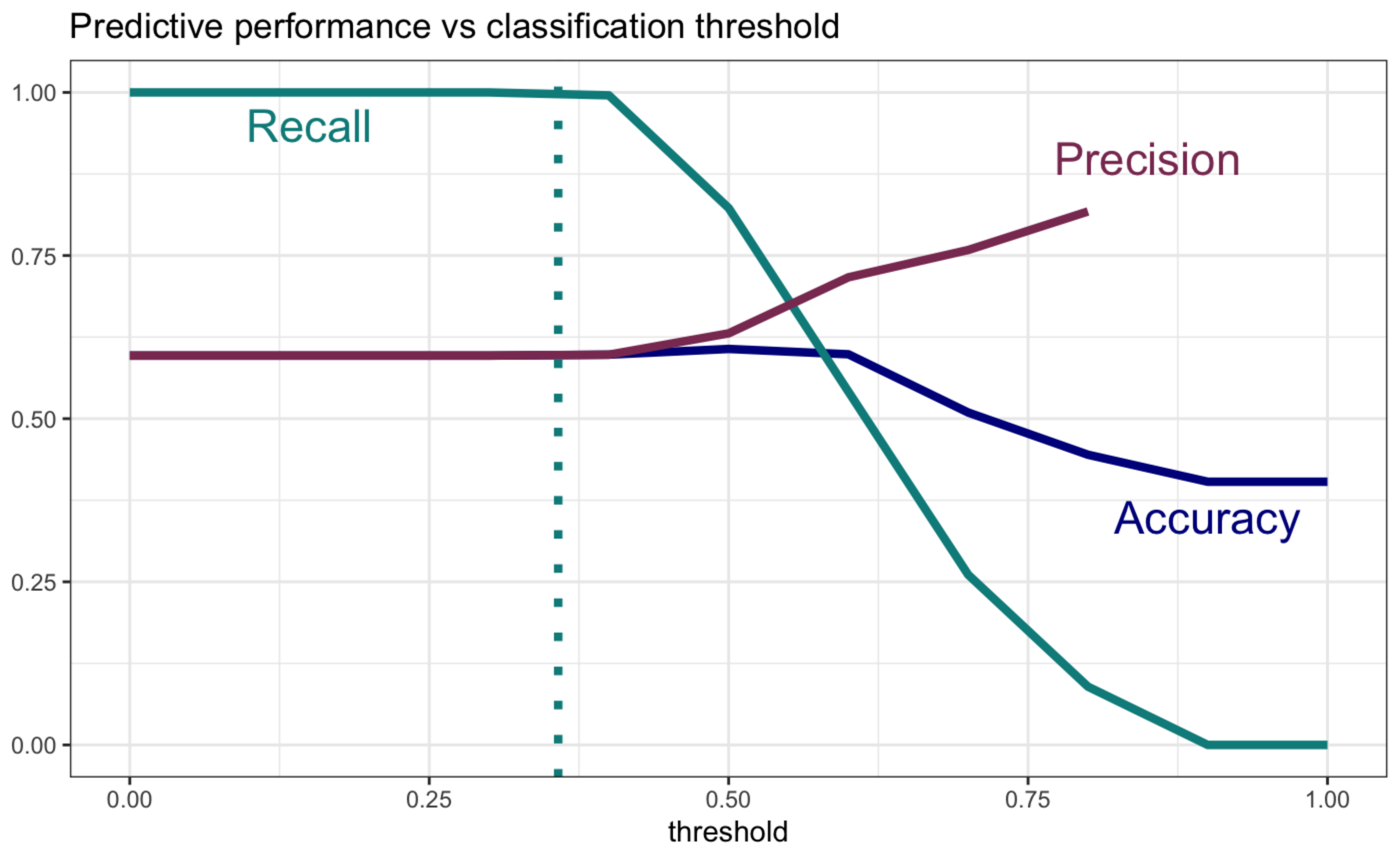
Intuition check

For low thresholds, recall is 1. For high thresholds, recall is 0. **Why?**

```
cces20 %>% select(.fitted) %>% summary()
```

```
##     .fitted
##  Min.   :0.3577
##  1st Qu.:0.5050
##  Median :0.5753
##  Mean   :0.5967
##  3rd Qu.:0.6808
##  Max.   :0.8492
```

Intuition check



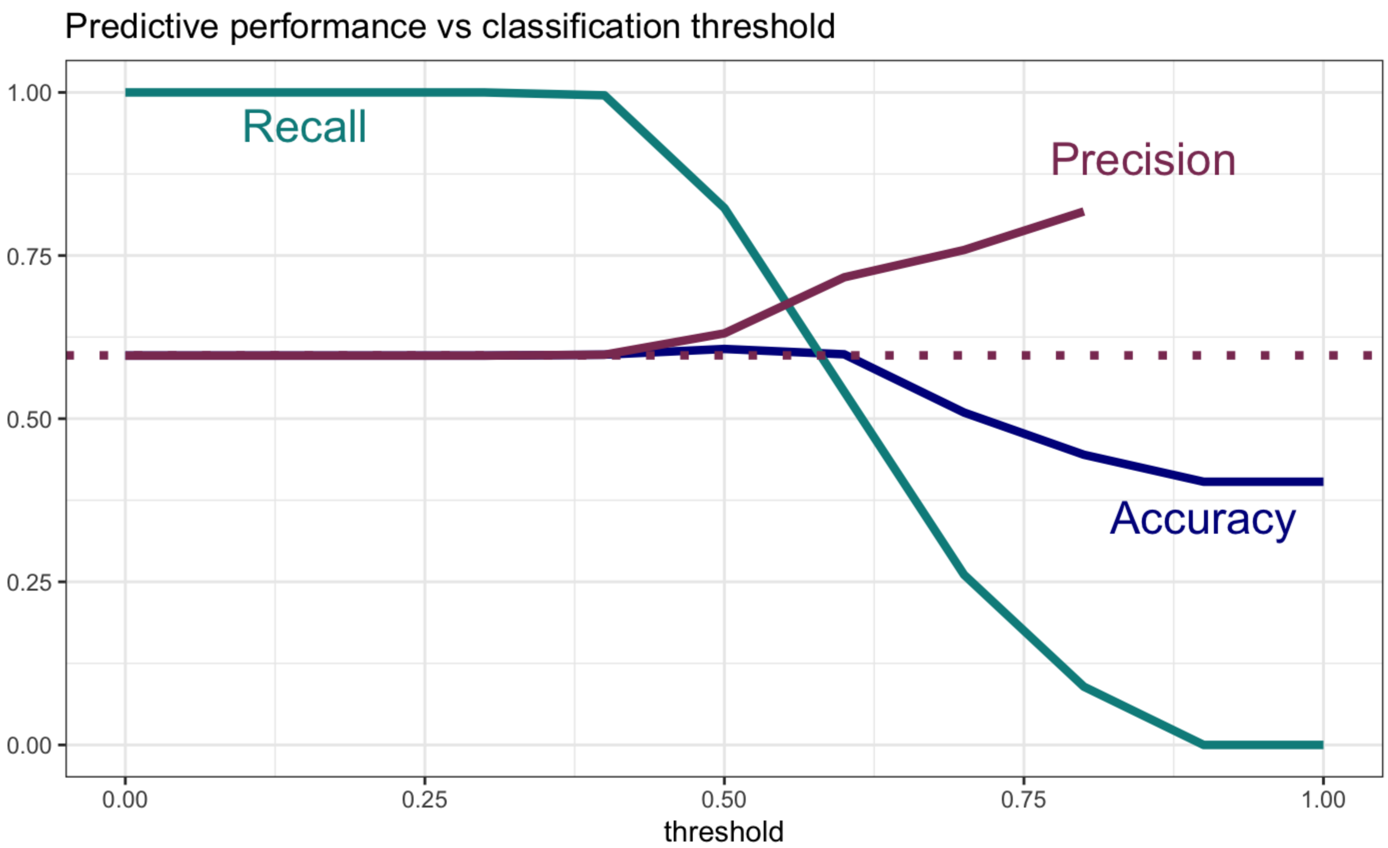
Intuition check

For low thresholds, accuracy and precision are constant at 0.597. **Why?**

```
cces20 %>%
  summarize(mean_y = mean(vote_dem))
```

```
## # A tibble: 1 × 1
##   mean_y
##    <dbl>
## 1  0.597
```

Intuition check



Predicting a numeric variable

Let's try to predict someone's *age* based on their marital status.

```
mod_age1 <- cces %>%
  lm(age ~ marstat,
     data = .)

tidy(mod_age1)
```

```
## # A tibble: 3 × 5
##   term           estimate std.error statistic  p.value
##   <chr>             <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)       57.5      0.205     281.  0
## 2 marstatMarried    -4.66     0.227     -20.6 9.36e-94
## 3 marstatSingle    -20.4      0.245     -83.1 0
```

Predicting a numeric variable

```
cces <- augment(mod_age1, cces) %>%
  select(age:.fitted) %>%
  mutate(error = .fitted - age)

head(cces)
```

```
## # A tibble: 6 × 7
##     age faminc marstat  ownhome pid3  .fitted  error
##   <dbl>  <dbl> <chr>    <fct>   <chr>   <dbl>  <dbl>
## 1    54  15000 Married  Rent    Rep      52.9  -1.13
## 2    65  65000 Divorced Rent    Dem      57.5  -7.47
## 3    58 110000 Married  Own     Dem      52.9  -5.13
## 4    53  15000 Single   Rent    Ind      37.1 -15.9
## 5    59  45000 Divorced Own     Rep      57.5  -1.47
## 6    50  55000 Married  Own     Rep      52.9   2.87
```

Measures of predictive accuracy

Root Mean Squared Error (RMSE)

$$\sqrt{\frac{1}{N}\sum_i (Y_i - \hat{Y}_i)^2}$$

Mean Absolute Error (MAE)

$$\frac{1}{N}\sum_i |Y_i - \hat{Y}_i|$$

Mean Absolute Percent Error (MAPE)

$$\frac{1}{N}\sum_i 100 \cdot \frac{|Y_i - \hat{Y}_i|}{Y_i}$$

Measures of predictive accuracy

We can calculate the RMSE, MAE, and MAPE manually:

```
cces %>%
  summarize(rmse = sqrt(sum(error^2) / nrow(cces)),
            mae = sum(abs(error)) / nrow(cces),
            mape = sum((abs(error)/age) * 100) / nrow(cces)
  )
```

```
## # A tibble: 1 × 3
##    rmse   mae  mape
##   <dbl> <dbl> <dbl>
## 1  15.0  12.5  30.7
```

Measures of predictive accuracy

Or we can use the canned functions `rmse()`, `mae()`, `mape()` in the {`tidymodels`} package:

```
cces %>%
  summarize(rmse = rmse(cces, truth = age, estimate = .fitted)$.estimate,
            mae = mae(cces, truth = age, estimate = .fitted)$.estimate,
            mape = mape(cces, truth = age, estimate = .fitted)$.estimate
  )
```

```
## # A tibble: 1 × 3
##    rmse   mae  mape
##   <dbl> <dbl> <dbl>
## 1  15.0  12.5  30.7
```

Improving our model

What if we include home ownership (`ownhome`) and household income (`faminc`) in our model too?

```
mod_age2 <- cces %>%
  lm(age ~ marstat + ownhome + faminc,
     data = .)

tidy(mod_age2)
```

```
## # A tibble: 5 × 5
##   term              estimate  std.error statistic   p.value
##   <chr>                <dbl>      <dbl>     <dbl>     <dbl>
## 1 (Intercept)     54.8       0.226          242.  0
## 2 marstatMarried  -5.74      0.233          -24.6 4.90e-133
## 3 marstatSingle  -19.5       0.240          -81.0 0
## 4 ownhomeOwn       7.55      0.169           44.7 0
## 5 faminc          -0.0000274 0.00000177     -15.5 2.90e- 54
```

Do you think the RMSE, MAE, MAPE will be smaller or larger?

Comparing two models

Initial model with just marital status as a predictor:

```
## # A tibble: 1 × 3
##    rmse   mae  mape
##   <dbl> <dbl> <dbl>
## 1  15.0  12.5  30.7
```

New model with home ownership and family income added:

```
## # A tibble: 1 × 3
##    rmse   mae  mape
##   <dbl> <dbl> <dbl>
## 1  14.6  12.2  29.7
```

Test vs training set

Of course, if we were really interested in the predictive power of our model we should look at how it performs on *new data* - i.e. data it was not trained on.

```
cces <-
  cces %>%
  mutate(test = sample(
    x = c(0, 1),
    size = nrow(cces),
    replace = TRUE
  ))

cces_split <- cces %>% group_by(test) %>% group_split()
cces_train <- cces_split[[1]]
cces_test <- cces_split[[2]]
```

Test vs training set

```
mod_age2 <- cces_train %>%
  lm(age ~ marstat + ownhome + faminc,
     data = .)

cces_test %>%
  mutate(.fitted = predict(mod_age2, .),
         error = .fitted - age) %>%
  summarize(
    rmse = rmse(., truth = age, estimate = .fitted)$.estimate,
    mae = mae(., truth = age, estimate = .fitted)$.estimate,
    mape = mape(., truth = age, estimate = .fitted)$.estimate
  )
```

```
## # A tibble: 1 × 3
##    rmse   mae  mape
##   <dbl> <dbl> <dbl>
## 1  14.6  12.2  29.7
```