Goodness of Fit and Predictive Power

Overview for Today

Today we will be learning about:

- Determining ways to understand how well your model fits your data
- Determining how well your model is able to predict your outcome

Continuing with Our Vote Behavior Model

```
# A tibble: 5,000 × 8
   vote  gender   age age_std age_std_sq life_vote vote_dist       vote_dist_num
   <chr> <chr>  <int>   <dbl>      <dbl> <chr>     <chr>                    <int>
 1 Y     M         71  1.79       3.19     N         Walkable - Mod…              1
 2 Y     M         46 -0.0243    0.000593 Y         Drive - Modera…              2
 3 Y     M         48  0.120     0.0145   Y         Walkable - Mod…              1
 4 N     M         62  1.13      1.29     Y         Walkable - Mod…              1
 5 Y     M         51  0.338     0.114    Y         Drive - Modera…              2
 6 N     M         18 -2.05      4.21     Y         Drive - Modera…              2
 7 N     F         35 -0.821     0.674    Y         Drive - Far                  3
 8 N     F         59  0.917     0.841    N         Drive - Modera…              2
 9 N     F         32 -1.04      1.08     N         Drive - Modera…              2
10 N     F         26 -1.47      2.17     Y         Drive - Modera…              2
# ℹ 4,990 more rows
```

What Are Goodness of Fit Tests?

Goodness of fit statistics (or fit statistics) and their tests are all ways to examine how well your model is able to reproduce your observed data.

Just remember:

“All models are wrong, but some are useful.” – George P. Box

Remember Maximum Likelihood?

Maximum likelihood is a method to find the right parameter estimates of our model.

It answers the question: what values should your parameter estimates (e.g. slope coefficients) take to **maximize the likelihood** of seeing your data?

Your Model's Log Likelihood: Lifetime Voter Status

Maximum likelihood ensures that the values of our parameter estimates are the values that maximize the likelihood of our data.

```
mod_lv <- glm(vote == "Y" ~ life_vote, data = data_vote,
              family = binomial(link = "logit"))

log_like_lv <- logLik(mod_lv)
```

```
[1] -3202.38
```

The Null Model & Its Likelihood

We can also fit different models like the Null Model—a model with no predictors—and obtain their likelihoods.

```
mod_null <- glm(vote == "Y" ~ 1, data = data_vote,
                family = binomial(link = "logit"))

log_like_null <- logLik(mod_null)
```

```
[1] -3465.51
```

Does Your Model Fit Better than the Null?

We can compare the fit of nested models using a **deviance test** which compares the log likelihood of the smaller (less complicated) model to the larger (more complicated) model.

A significant **deviance test** means the more complicated model fits our data better than the less complicated model.

```
deviance <- 2 * (logLik(mod_lv) - logLik(mod_null))
pchisq(deviance, df = 1, lower.tail = F)
```

```
'log Lik.' 1.844343e-116 (df=2)
```

```
[1] "Deviance = 526.25"
```

```
[1] "p = 0"
```

Nested Model Comparisons

We can use the **deviance test** to compare any set of nested models. A nested set of models is where the smaller model (model with less predictors) is a trimmed down version of the larger model (model with more predictors) that has been estimated from the exact same data.

```
mod_large <- glm(vote == "Y" ~ gender + age_std + age_std_sq + life_vote * vot
                 data = data_vote, family = binomial)

mod_small <- glm(vote == "Y" ~ gender + age_std + life_vote + vote_dist,
                 data = data_vote, family = binomial)
```

Nested Model Comparisons

If the deviance test is significant, it means that the larger model fits your data significantly better than the smaller model and should be selected over the smaller model.

```
1 anova(mod_small, mod_large, test = "Chisq")
```

```
Analysis of Deviance Table

Model 1: vote == "Y" ~ gender + age_std + life_vote + vote_dist
Model 2: vote == "Y" ~ gender + age_std + age_std_sq + life_vote * vote_dist
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1      4993     6352.2
2      4989     6237.0  4   115.25 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Pseudo-R-Squareds

Log likelihoods can also be used to calculate Pseudo-R-Squared values, which are similar (although not identical) to the linear regression R^2.

There are a handful of Pseudo-R-Squareds to choose from, but I usually go with McFadden's R-Squared.

McFadden's R-Squared

McFadden's R-Squared tells us how much of an improvement our target model is over our null model.

$$R^2 = \frac{LL_{Null} - LL_{Model}}{LL_{Null}} = 1 - \frac{LL_{Model}}{LL_{Null}}$$

```
1  1 - logLik(mod_large) / logLik(mod_null)
```

```
[1] "McFadden: = 0.1"
```

Using Predicted Probability to Assess Model Fit

We can use our models' predicted probabilities to assess how well our model fits our data. If our model fits our data well, then our predicted probabilities should be higher for outcomes where the value equals 1 compared to outcomes where the value equals 0.

Correlation Between the Outcome and Model Predictions

The easiest and quickest way to see how related the predicted probably values are to the outcome is to correlate the two. Higher correlations indicate a stronger relationship and better fit between your model and data.

```
predicted_values <- fitted(mod_large)

cor(data_vote$vote == "Y", predicted_values)
```

```
[1] "R = 0.36"
```

From Predicted Probability to Predicted Outcome

We can transform our predicted probability into a predicted outcome by setting a threshold value. For instance, we can say that a predicted probability value greater than .50 becomes a 1 and a predicted probability value less than or equal to .50 becomes a 0.

We can then create a classification table:

```r
data_vote <-
  data_vote |>
  dplyr::mutate(
    pred_vote = dplyr::if_else(fitted(mod_large) > .50, "Y", "N")
  )

xtabs(~vote + pred_vote, data_vote) |> addmargins()
```

```
      pred_vote
vote      N    Y  Sum
  N    1167 1309 2476
  Y     462 2062 2524
  Sum  1629 3371 5000
```

True Positive & False Positive Rates

Using the classification table, we can calculate the true positive rate (sensitivity), the true negative rate (specificity), and the overall model accuracy:

$$TP = P(\hat{Y} = 1 | Y = 1)$$

$$TN = P(\hat{Y} = 0 | Y = 0)$$

$$Acc. = TP \times P(Y = 1) + TN \times P(Y = 0)$$

True Positive & False Positive Rates

```
1 xtabs(~vote + pred_vote, data_vote) |> prop.table(1) |> round(2)
```

```
    pred_vote
vote    N    Y
   N 0.47 0.53
   Y 0.18 0.82
```

```
1 xtabs(~vote + pred_vote, data_vote) |> prop.table() |> round(2)
```

```
    pred_vote
vote    N    Y
   N 0.23 0.26
   Y 0.09 0.41
```

From the first table, we can see the true positive rate is 0.82 and the true negative rate is 0.47.

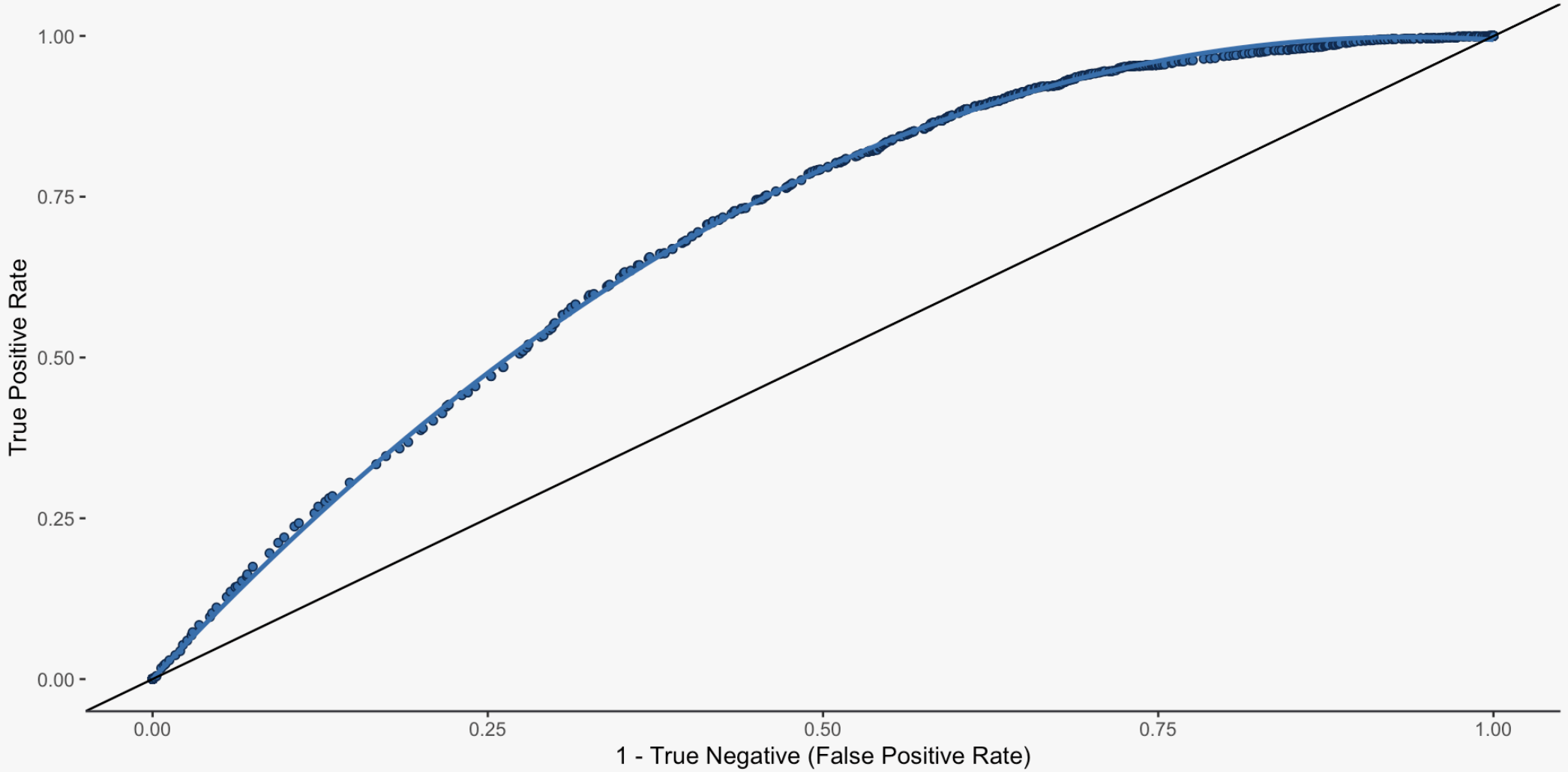
From the second table, we can see the accuracy is 0.41 plus 0.23, which equals 0.64.

Receiver Operator Characteristic Curve: ROC Curve

The predicted outcome threshold (.50 in our example) is arbitrary. It is possible that a different choice of threshold would give us a better true positive and true negative rate.

To explore how our true positive and negative rates change depending on our threshold, we can create a plot called a **receiver operating characteristic (ROC) curve**.

Receiver Operator Characteristic Curve: ROC Curve



Area Under the Curve: AUC

The area underneath the ROC curve or AUC gives us a summary metric for how well our model is able to balance the true positive rate and the true negative rate.

AUC is always going to be between 1 (perfect prediction) and .50 (predicting by a coin toss).

AUC Value	Interpretation
AUC = .50	No discrim.
$.50 < AUC < .70$	Poor discrim.
$.70 \leq AUC < .80$	Accept. discrim.
$.80 \leq AUC < .90$	Excellent discrim.
$AUC \geq .90$	Outstanding discrim.

Using R to Calculate AUC

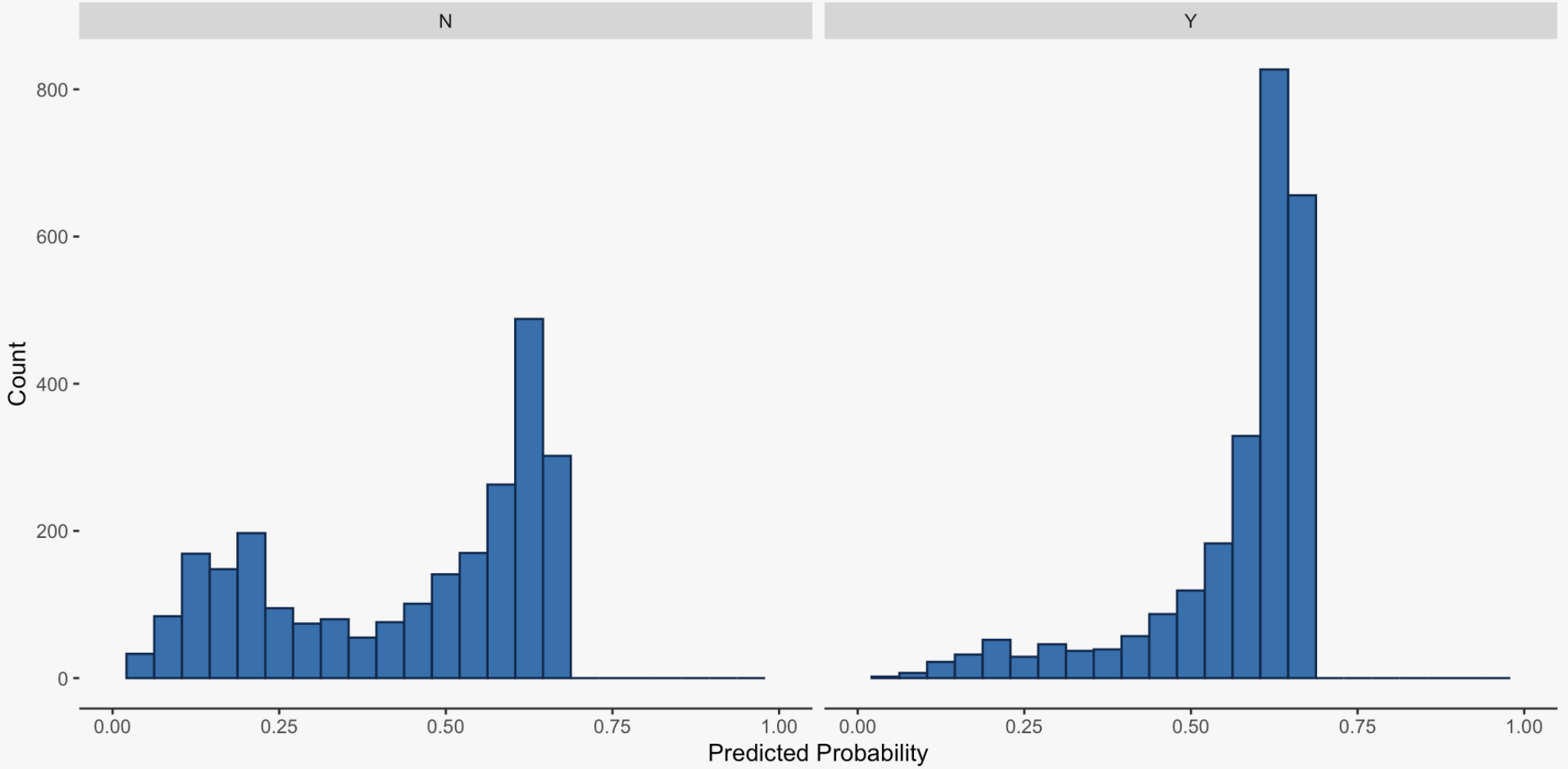
```
1  pROC::roc(vote == "Y" ~ fitted(mod_large), data = data_vote)
```

```
Call:
roc.formula(formula = vote == "Y" ~ fitted(mod_large), data = data_vote)

Data: fitted(mod_large) in 2476 controls (vote == "Y" FALSE) < 2524 cases (vote == "Y"
TRUE).
Area under the curve: 0.6938
```

Plotting Predicted Probabilites by Outcome Class

It is also helpful to plot a histogram of the predicted probabilities by outcome to see how well your model is able to discriminate from success (1) and failures (0).



Examples of Bad, OK, and Good AUC Values