Lesson 14: Model Building

With an emphasis on prediction

Nicky Wakim

2025-05-14

Learning Objectives

1. Understand the place of LASSO regression within association and prediction modeling for binary outcomes.
2. Recognize the process for `tidymodels`
3. Understand how penalized regression is a form of model/variable selection.
4. Perform LASSO regression on a dataset using R and the general process for classification methods.

Keep in mind:

- We will be introducing another coding technique to fit models
- We will be introducing another way to build models

Learning Objectives

1. Understand the place of LASSO regression within association and prediction modeling for binary outcomes.
2. Recognize the process for `tidymodels`
3. Understand how penalized regression is a form of model/variable selection.
4. Perform LASSO regression on a dataset using R and the general process for classification methods.

Some important definitions

- **Model selection**: picking the "best" model from a set of possible models
  - Models will have the same outcome, but typically differ by the covariates that are included, their transformations, and their interactions
  - "Best" model is defined by the research question and by how you want to answer it!

- **Model selection strategies**: a process or framework that helps us pick our "best" model
  - These strategies often differ by the approach and criteria used to the determine the "best" model

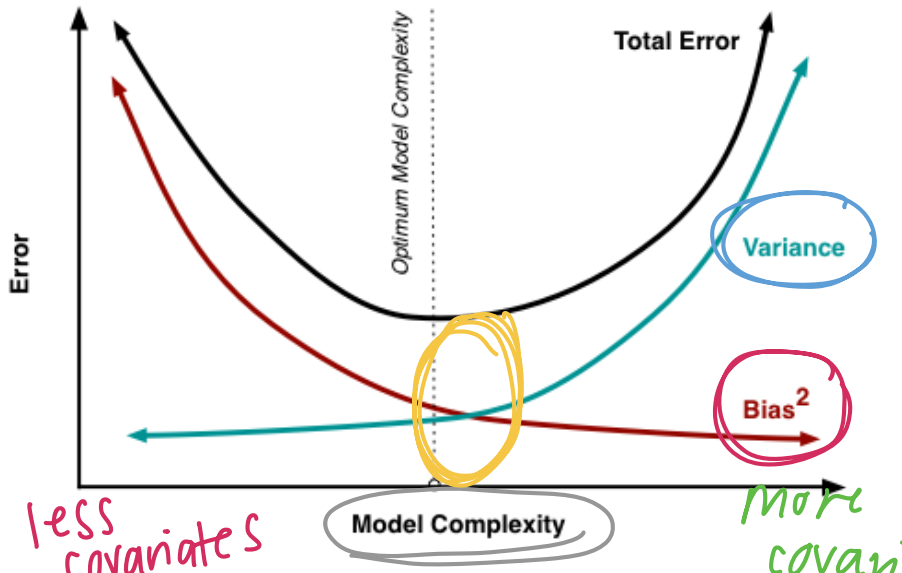
- **Overfitting**: result of fitting a model so closely to our *particular* sample data that it cannot be generalized to other samples (or the population)

Bias-variance trade off

- Recall from 512/612: MSE can be written as a function of the bias and variance

$$MSE = \text{bias}\left(\widehat{\beta}\right)^2 + \text{variance}\left(\widehat{\beta}\right)$$

  - **We no longer use MSE in logistic regression to find the best fit model, BUT the idea between the bias and variance trade off holds!**
- For the same data:
  - More covariates in model: less bias, more variance
    - Potential overfitting: with new data does our model still hold?
  - Less covariates in model: more bias, less variance
    - More bias bc more likely that were are not capturing the true underlying relationship with less variables

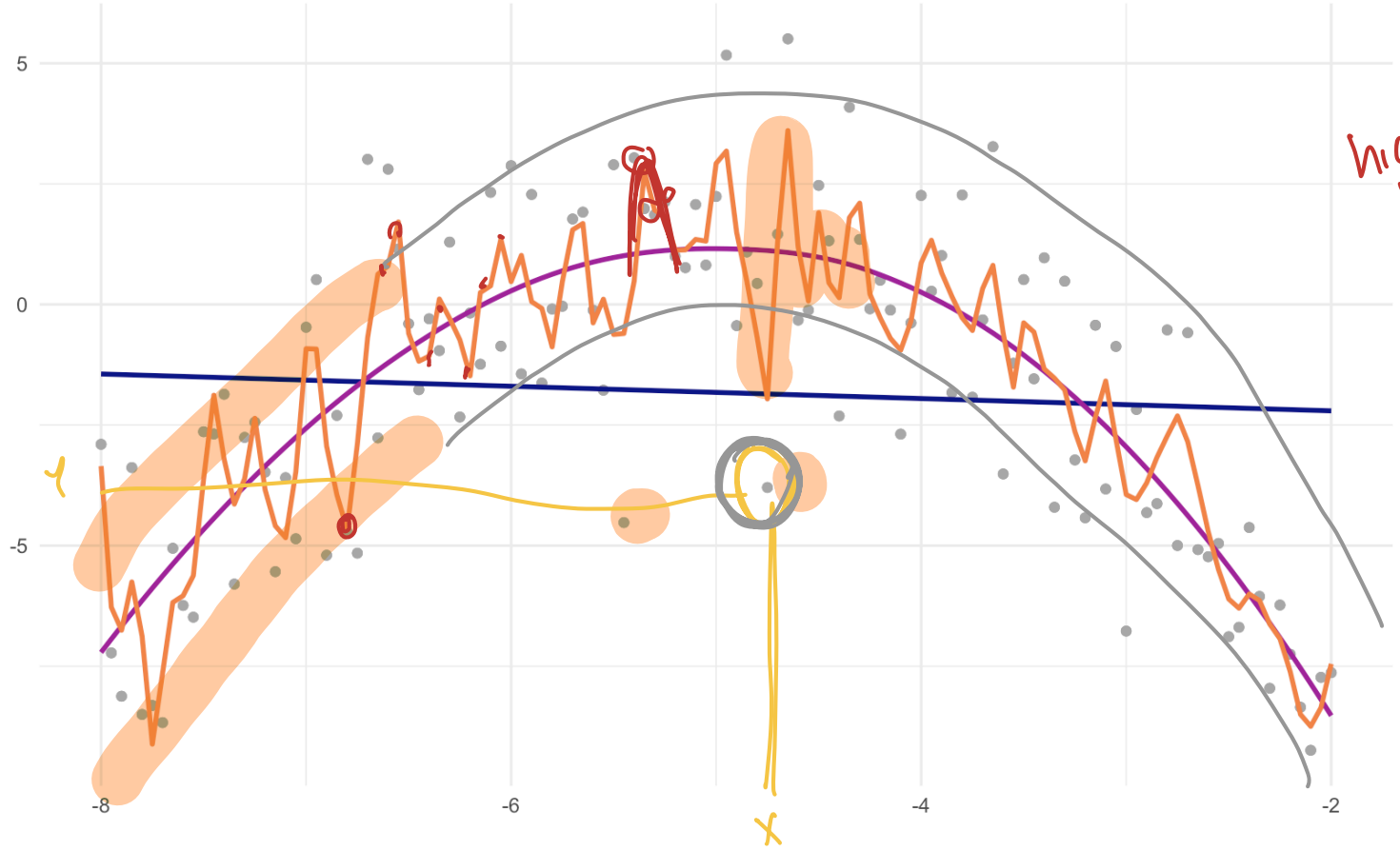


Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

increasing # covariates in model

Visual of overfitting and underfitting data

From Data Science in a Box:



higher bias → not capturing true underlying relation.

less var

linear regression

less bias, more variance

The goals of association vs. prediction

less bias, more var

Association / Explanatory / One variable's effect

- **Goal:** Understand one variable's (or a group of variable's) effect on the response after adjusting for other factors
- Mainly interpret odds ratios of the variable that is the focus of the study

Relationship b/w vars

RQ: How is foodinsecurity associated w/ household income?

Prediction

- **Goal:** to calculate the most precise prediction of the response variable
- Interpreting coefficients is not important
- Choose only the variables that are strong predictors of the response variable
  - Excluding irrelevant variables can help reduce widths of the prediction intervals

RQ: How well can we predict food insecurity? What are the biggest contributing factors to prediction?

Model selection strategies for *categorical* outcomes

Association / Explanatory / One variable's effect

- Selection of potential models is tied more with the research context with some incorporation of prediction scores

---

- Pre-specification of multivariable model
- Purposeful model selection
  - "Risk factor modeling"
- Change in Estimate (CIE) approaches
  - Will learn in ~~Survival~~ Analysis (BSTA 514)

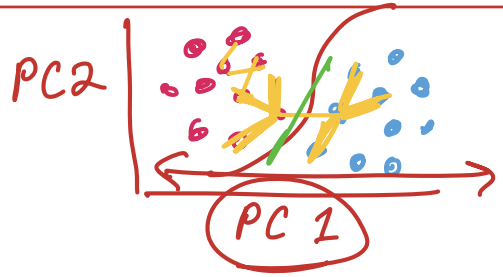
time to event

Prediction

- Selection of potential models is fully dependent on prediction scores

---

- Logistic regression with more refined model selection
  - Regularization techniques (LASSO, Ridge, Elastic net)
- Machine learning realm — not so model based
  - Decision trees, random forest, k-nearest neighbors, Neural networks



Before I move on...

- We CAN use purposeful selection from last quarter in **any** type of generalized linear model (GLM)
  - This includes logistic regression!

- The best documented information on purposeful selection is in the Hosmer-Lemeshow textbook on logistic regression
  - Textbook in student files is linked here
  - Purposeful selection starts on page 89 (or page 101 in the pdf)

- I will not discuss purposeful selection in this course
  - Be aware that this is a tool that you can use in any regression!

Okay, so prediction of categorical outcomes

- **Classification:** process of predicting categorical responses/outcomes
  - Assigning a category outcome based on an observation's predictors

- Note: we've already done a lot of work around predicting probabilities within logistic regression
  - Can we take those predicted probabilities one step further to predict the binary outcome??

- Common classification methods (good site on brief explanation of each)
  - Logistic regression
  - Naive Bayes
  - k-Nearest Neighbor (KNN)
  - Decision Trees
  - Support Vector Machines (SVMs)
  - Neural Networks

Logistic regression is a classification method

- But to be a **good classifier**, our logistic regression model needs to be **built a certain way**

- Prediction depends on type of variable/model selection!
  - This is when it can become machine learning

- So the big question is: how do we select this model??
  - Regularized techniques, aka penalized regression

Poll Everywhere Question 1

association, centered on relationship b/w outcome & predictors

PREDICTION - can you pred outcome?

Learning Objectives

1. Understand the place of LASSO regression within association and prediction modeling for binary outcomes.

2. Recognize the process for `tidymodels`

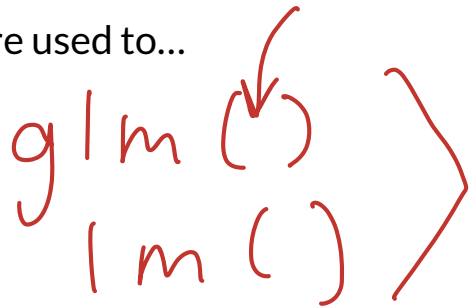
3. Understand how penalized regression is a form of model/variable selection.

4. Perform LASSO regression on a dataset using R and the general process for classification methods.

Before I get really into things!!

- `tidymodels` is a great package when we are performing prediction

- One problem: it uses very different syntax for model fitting than we are used to...



- `tidymodels` syntax dictates that we need to define:
  - A model
  - A recipe
  - A workflow

tidymodels with GLOW

glm = assoc
glm net = prediction

To fit our logistic regression model with the interaction between age and prior fracture, we use:

- **Model:** saves the type of regression that we will use (we will be performing logistic regression)

```
model = logistic_reg()
```

Similar to specifying family in glm()

- **Recipe:** sets the formula for the regression model
  - Specify categorical variables with `step_dummy()`
  - Specify interactions with `step_interactions()`

```
recipe = recipe(fracture ~ priorfrac + age_c, data = glow1) %>%
         step_dummy(priorfrac) %>%
         step_interact(terms = ~ age_c:starts_with("priorfrac"))
```

I(PF="Yes")

dummy variable

- **Workflow:** step through the model and recipe
  - Similar to the single step taken in `glm()`

```
workflow = workflow() %>% add_model(model) %>% add_recipe(recipe)
```

- **Fit:** specify the data with which we will fit the model

```
fit = workflow %>% fit(data = glow1)
```

glm() engine
will fit model

- run on own
- run through tidymodels

glmnet() engine
will fit a LASSO model

- run on own
- BETTER through tidymodels

syn 1

```
data %>% mutate(color = "red")
```

syn 2

```
data$color = "red"
```

↳ if shade = pink,
color = red

tidymodels with GLOW: Results

- Print the results from our fitted model using `tidymodels`

```
1 tidy(fit, conf.int = T) %>% gt() %>%
2   tab_options(table.font.size = 35) %>%
3   fmt_number(decimals = 3)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	−1.376	0.134	−10.270	0.000	−1.646	−1.120
age_c	0.063	0.015	4.043	0.000	0.032	0.093
priorfrac_Yes	1.002	0.240	4.184	0.000	0.530	1.471
age_c_x_priorfrac_Yes	−0.057	0.025	−2.294	0.022	−0.107	−0.008

Same as results from previous lessons

```
glow_m3 = glm(fracture ~ priorfrac + age_c + priorfrac*age_c,
              data = glow1, family = binomial)
```

```
tidy(glow_m3, conf.int = T) %>% gt() %>%
  tab_options(table.font.size = 35) %>%
  fmt_number(decimals = 3)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	−1.376	0.134	−10.270	0.000	−1.646	−1.120
priorfracYes	1.002	0.240	4.184	0.000	0.530	1.471
age_c	0.063	0.015	4.043	0.000	0.032	0.093
priorfracYes:age_c	−0.057	0.025	−2.294	0.022	−0.107	−0.008

Interaction model:

$$\text{logit}\left(\widehat{\pi}(\mathbf{X})\right) = \widehat{\beta}_0 + \widehat{\beta}_1 \cdot I(\text{PF}) + \widehat{\beta}_2 \cdot Age + \widehat{\beta}_3 \cdot I(\text{PF}) \cdot Age$$

$$\text{logit}\left(\widehat{\pi}(\mathbf{X})\right) = -1.376 + 1.002 \cdot I(\text{PF}) + 0.063 \cdot Age - 0.057 \cdot I(\text{PF}) \cdot Age$$

- Reminder of main effects and interactions

Learning Objectives

1. Understand the place of LASSO regression within association and prediction modeling for binary outcomes.
2. Recognize the process for `tidymodels`
3. Understand how penalized regression is a form of model/variable selection.
4. Perform LASSO regression on a dataset using R and the general process for classification methods.

Penalized regression

- **Basic idea:** We are running regression, but now we want to **incentivize our model fit to have less predictors**
  - Include a **penalty to discourage too many predictors** in the model

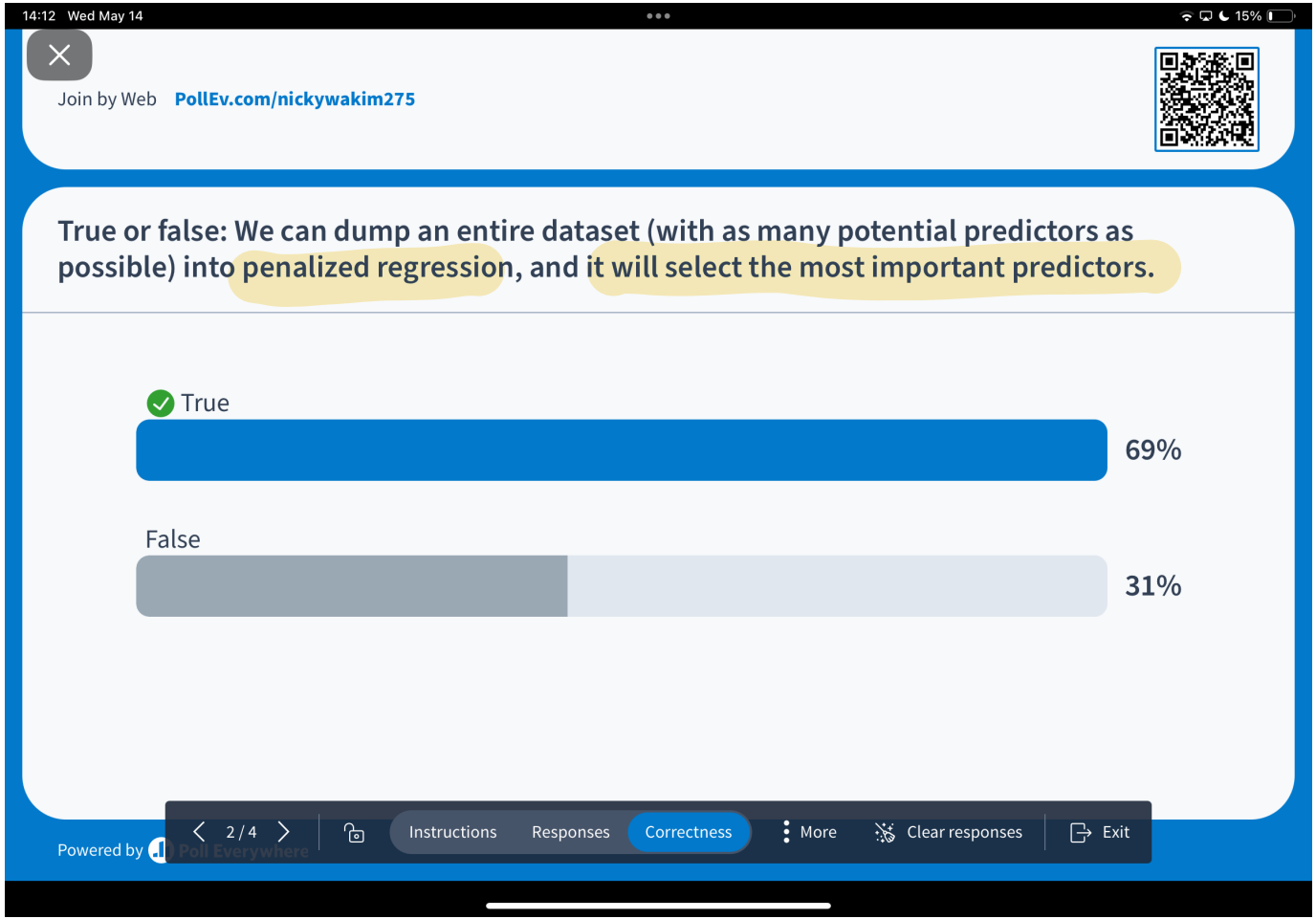
- Also known as *shrinkage* or *regularization* methods
  - "Shrinks" coefficient estimates to 0

- Penalty will reduce coefficient values to zero (or close to zero) if the predictor does not contribute much information to predicting our outcome

covariate

- We need a tuning parameter that determines the amount of shrinkage called lambda/λ
  - How much do we want to penalize additional predictors?

Poll Everywhere Question 2



Three types of penalized regression

Main difference is the type of penalty used

Ridge regression

- Penalty called L2 norm, uses squared values
- Pros
  - Reduces overfitting
  - Handles $p > n$
  - Handles collinearity
- Cons
  - Does not shrink coefficients to 0
  - Difficult to interpret

Lasso regression

- Penalty called L1 norm, uses absolute values
- Pros
  - Reduces overfitting
  - Shrinks coefficients to 0
- Cons
  - Cannot handle $p > n$
  - Does not handle multicollinearity well

Elastic net regression

- L1 and L2 used, best of both worlds
- Pros
  - Reduces overfitting
  - Handles $p > n$
  - Handles collinearity
  - Shrinks coefficients to 0
- Cons
  - More difficult to do than other two

Learning Objectives

1. Understand the place of LASSO regression within association and prediction modeling for binary outcomes.
2. Recognize the process for `tidymodels`
3. Understand how penalized regression is a form of model/variable selection.

4. Perform LASSO regression on a dataset using R and the general process for classification methods.

Overview of predicton model building steps

1. Split data into **training** and **testing** datasets

2. Perform our classification method on **training set**
   - This is where we will use penalized regression!

build model or classifier w/ training set

3. Measure predictive accuracy on **testing set**

Example to be used: GLOW Study

- From GLOW (Global Longitudinal Study of Osteoporosis in Women) study

- **Outcome variable:** any fracture in the first year of follow up (FRACTURE: 0 or 1)

- ~~**Risk factor/variable of interest:** history of prior fracture (PRIORFRAC: 0 or 1)~~
- ~~**Potential confounder or effect modifier:** age (AGE, a continuous variable)~~
  - ~~Center age will be used! We will center around the rounded mean age of 69 years old~~

- Crossed out because we are no longer attached to specific predictors and their association with fracture
  - Focused on **predicting fracture with whatever variables we can!**

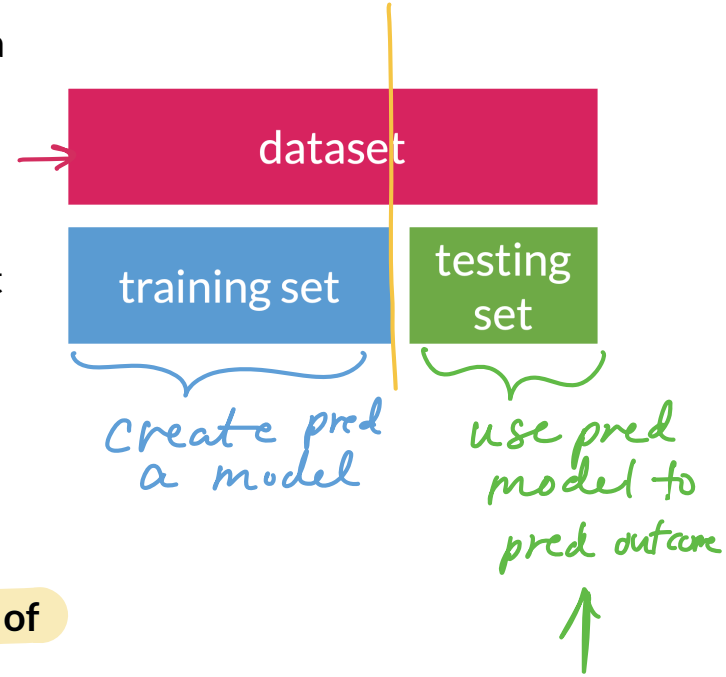
Step 1: Splitting data

- **Training:** act of creating our prediction model based on our observed data
  - Supervised: Means we keep information on our outcome while training

- **Testing:** act of measuring the predictive accuracy of our model by trying it out on *new* data

- When we use data to create a prediction model, we want to test our prediction model on *new* data
  - Helps make sure prediction model can be applied to other data **outside of the data that was used to create it!**

- So an important first step in prediction modeling is to *split our data* into a **training set** and a **testing set**!



Step 1: Splitting data

Training set

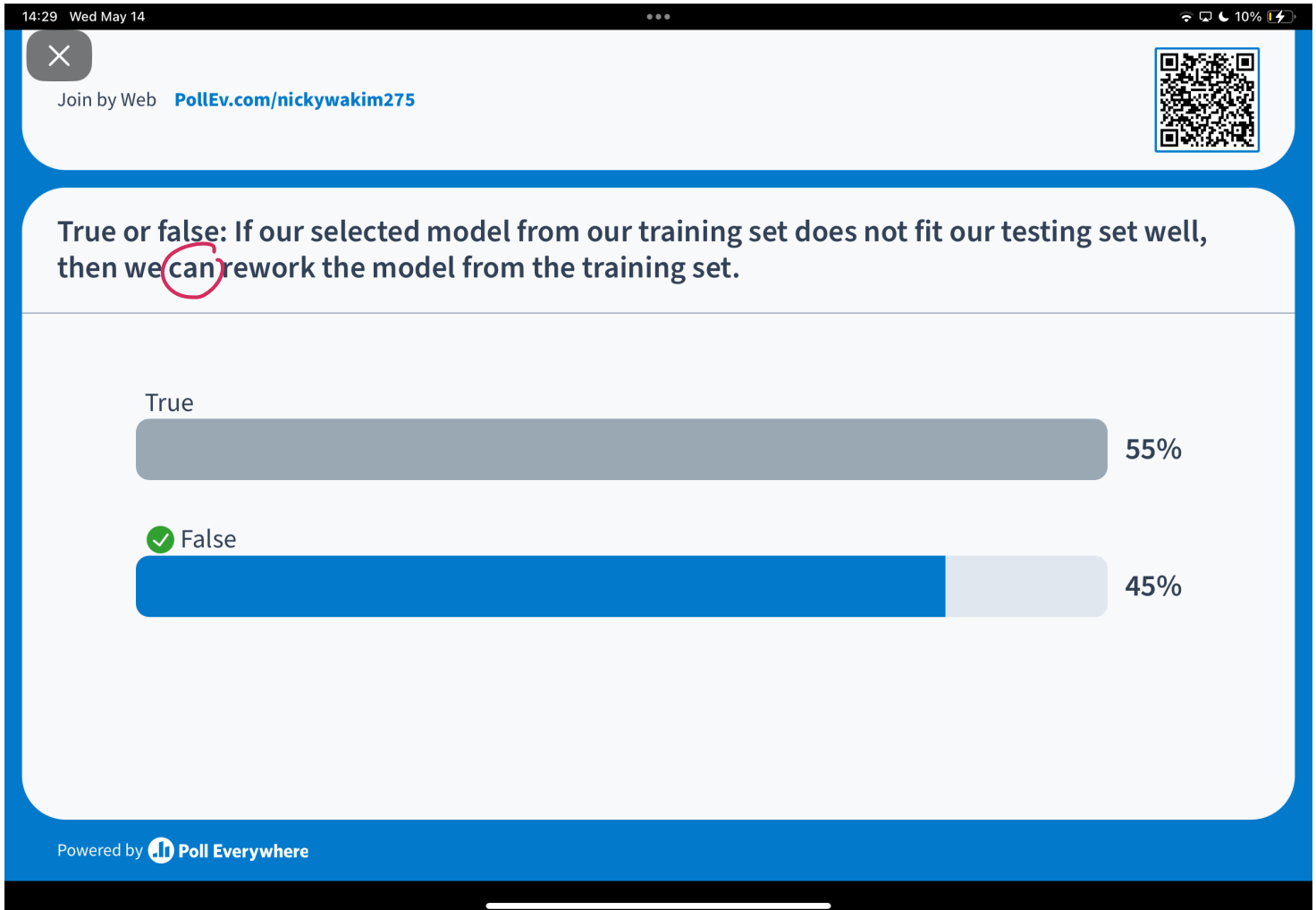
- Sandbox for model building
- Spend most of your time using the training set to develop the model
- Majority of the data (usually 80%)

Testing set

- Held in reserve to determine efficacy of one or two chosen models
- Critical to look at it once at the end, otherwise it becomes part of the modeling process
- Remainder of the data (usually 20%)

- Slide content from Data Science in a Box

Poll Everywhere Question 3

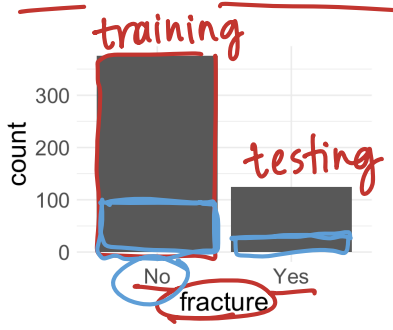


Step 1: Splitting data

- When splitting data, we need to be conscious of the proportions of our outcomes
  - Is there imbalance within our outcome?
  - We want to randomly select observations but make sure the proportions of No and Yes stay the same
  - We **stratify** by the outcome, meaning we pick Yes's and No's separately for the training set

b/w training & testing

```
1 ggplot(glow1, aes(x = fracture)) + geom_bar()
```



- Side note: took out `bmi` and `weight` bc we have multicollinearity issues
  - Combo of I hate these variables and my previous work in the LASSO identified these as not important

```
1 glow = glow1 %>%
2     dplyr::select(-sub_id, -site_id, -phy_id, -age, -bmi, -weight)
```

↳ age_c

Step 1: Splitting data

- From package `rsample` within `tidyverse`, we can use `initial_split()` to create training and testing data
  - Use `strata` to stratify by fracture
  - Use `prop` to set the proportion of training data

80% training

```
glow_split = initial_split(glow, strata = fracture, prop = 0.8)
glow_split
```

```
<Training/Testing/Total>
<400/100/500>
```

train. test & total

- Then we can pull the training and testing data into their own datasets

```
glow_train = training(glow_split)
glow_test = testing(glow_split)
```

400 obs: 300 No, 100 Yes

100 obs: 75 No, 25 yes

result of strata set to frac.

Step 1: Splitting data: peek at the split

```r
glimpse(glow_train)
```

```
Rows: 400
Columns: 10
$ priorfrac <fct> No, No, Yes, No, No, Yes, No, Yes, Yes, No, No, No,
No, No, …
$ height    <int> 158, 160, 157, 160, 152, 161, 150, 153, 156, 166,
153, 160, …
$ premeno   <fct> No, No, No, No, No, No, No, No, No, No, No, Yes,
No, No, No,…
$ momfrac   <fct> No, No, Yes, No, No, No, No, No, No, No, Yes, No,
No, No, No…
$ armassist <fct> No, No, Yes, No, No, No, No, No, No, No, No, No,
Yes, No, No…
$ smoke     <fct> No, No, No, No, No, Yes, No, No, No, No, Yes, No,
No, No, No…
$ raterisk  <fct> Same, Same, Less, Less, Same, Same, Less, Same,
Same, Less, …
$ fracscore <int> 1, 2, 11, 5, 1, 4, 6, 7, 7, 0, 4, 1, 4, 2, 2, 7, 2,
1, 4, 5,…
$ fracture  <fct> No, No, No, No, No, No, No, No, No, No, No, No, No,
No, No, …
$ age_c     <dbl> -7, -4, 19, 13, -8, -2, 15, 13, 17, -11, -2, -5,
-1, -2, 0, …
```

```r
glimpse(glow_test)
```

```
Rows: 100
Columns: 10
$ priorfrac <fct> No, No, No, No, No, No, No, No, Yes, Yes, No, No,
No, No, No…
$ height    <int> 167, 162, 165, 158, 153, 170, 154, 171, 142, 152,
166, 154, …
$ premeno   <fct> No, No, No, Yes, No, Yes, Yes, Yes, Yes, No, No,
No, No, No,…
$ momfrac   <fct> No, No, No, No, No, Yes, No, No, Yes, No, No, No,
No, No, No…
$ armassist <fct> Yes, No, Yes, No, Yes, No, Yes, No, No, No, No, No,
No, No, …
$ smoke     <fct> Yes, Yes, No, No, No, No, No, No, No, No, No, No,
No, No, No…
$ raterisk  <fct> Same, Less, Less, Greater, Same, Same, Same, Same,
Same, Sam…
$ fracscore <int> 3, 1, 5, 1, 8, 3, 7, 1, 6, 7, 0, 2, 0, 0, 1, 2, 2,
8, 4, 3, …
$ fracture  <fct> No, No, No, No, No, No, No, No, No, No, No, No, No,
No, No, …
$ age_c     <dbl> -13, -10, 3, -8, 17, 0, 6, -5, 1, 17, -11, -6, -10,
-12, -6,…
```

Step 2: Fit LASSO penalized logistic regression model

- Using Lasso penalized regression!
- We can simply set up a penalized regression model

model

```
lasso_mod = logistic_reg(penalty = 0.002, mixture = 1) %>%

            set_engine("glmnet")
```

- `glmnet` takes the basic fitting of `glm` and adds penalties!
  - In `tidymodels` we set an engine that will fit the model
- `mixture` option let's us pick the penalty
  - `mixture = 0` for Ridge regression
  - `mixture = 1` for Lasso regression
  - `0 < mixture < 1` for Elastic net regression

Step 2: Fit LASSO: Main effects

- Set the recipe so that we include all possible variables (from `fracture ~ .`)
- Set `step_dummy()` so R identifies categorical variables

```
glow_rec_main = recipe(fracture ~ ., data = glow_train) %>%

  step_dummy(priorfrac, premeno, momfrac, armassist, smoke, raterisk)
```

all categorical (binary & multi-level)

- We can add the LASSO model we already constructed
- Then we fit the model with the training data

```
glow_workflow_main = workflow() %>%

      add_model(lasso_mod) %>% add_recipe(glow_rec_main)

glow_fit_main = glow_workflow_main %>% fit(glow_train)
```

Step 2: Fit LASSO: Main effects: Identify variables

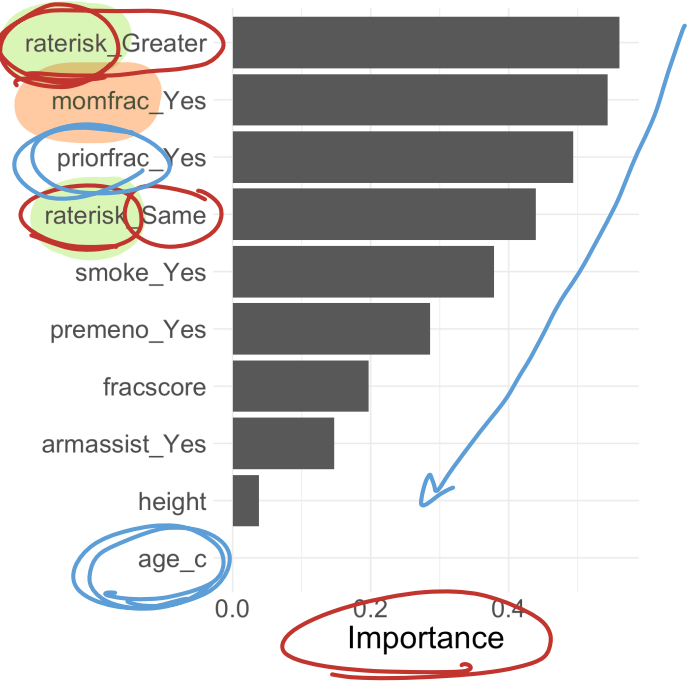
```r
library(vip)
vi_data_main = glow_fit_main %>%
    pull_workflow_fit() %>%
    vi(lambda = 0.002) %>% # vi: variable i
    filter(Importance != 0)
vi_data_main
```

```
# A tibble: 9 × 3
  Variable          Importance Sign
  <chr>                  <dbl> <chr>
1 raterisk_Greater       0.535  POS
2 momfrac_Yes            0.526  POS
3 priorfrac_Yes          0.485  POS
4 raterisk_Same          0.413  POS
5 smoke_Yes              0.344  NEG
6 premeno_Yes            0.267  POS
7 fracscore              0.196  POS
8 armassist_Yes          0.138  POS
9 height                 0.0370 NEG
```

importance

What exactly does this mean?

```r
glow_fit_main %>%
  extract_fit_parsnip() %>%
  vip(num_features = 20) + theme(text = ele
```



- Looks like age is removed!

Step 2: Fit LASSO: Look at model fit

```
glow_fit_main %>% tidy() %>% gt() %>%
  tab_options(table.font.size = 35) %>%
  fmt_number(decimals = 3)
```

term	estimate	penalty
(Intercept)	3.417	0.002
height	−0.037	0.002
fracscore	0.196	0.002
age_c	0.000	0.002
priorfrac_Yes	0.485	0.002
premeno_Yes	0.267	0.002
momfrac_Yes	0.526	0.002
armassist_Yes	0.138	0.002
smoke_Yes	−0.344	0.002
raterisk_Same	0.413	0.002
raterisk_Greater	0.535	0.002

Poll Everywhere Question 4

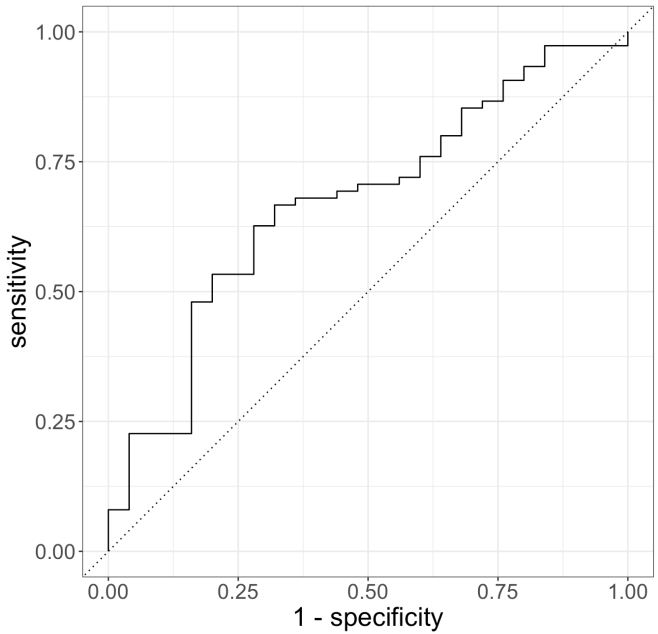
Step 3: Prediction on testing set

```r
glow_test_pred = predict(glow_fit_main, new_data = glow_test, type = "prob") %>%
    bind_cols(glow_test)
```

```r
glow_test_pred %>%
  roc_auc(truth = fracture,
                  .pred_No)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.672
```

```r
glow_test_pred %>%
  roc_curve(truth = fracture, .pred_No) %>%
  autoplot() + theme(text = element_text(size=20
```



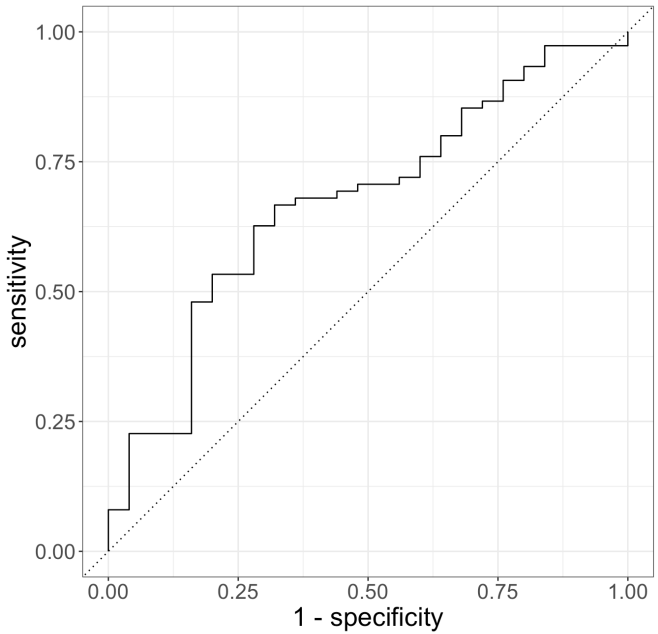
Step 3: Prediction on testing set

```
glow_test_pred = predict(glow_fit_main, new_data = glow_test, type = "prob") %>%
    bind_cols(glow_test)
```

```
glow_test_pred %>%
  roc_auc(truth = fracture,
                  .pred_No)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.672
```

```
glow_test_pred %>%
  roc_curve(truth = fracture, .pred_No) %>%
  autoplot() + theme(text = element_text(size=20
```

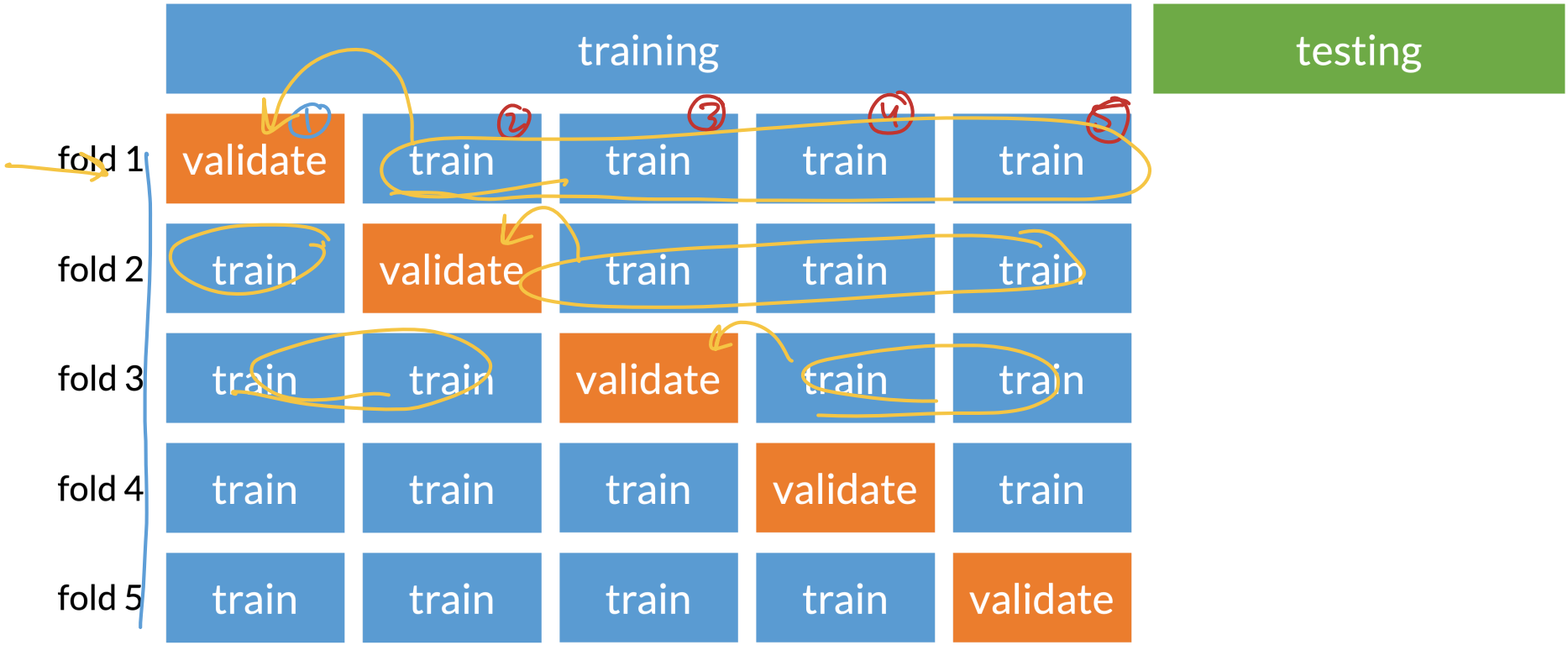


Why is this AUC worse than the one we saw with prior fracture, age, and their interaction?

- Only 1 training and testing set: can overfit training and perform poorly on testing
- We did not tune our penalty
- Our testing set only has 100 observations!

Cross-validation (specifically k-fold)

- Prevents overfitting to one set of training data
- Split data into folds that train and validate model selection
- Basically subsection of training and testing (called validating) before truly testing on our original testing set



Solutions / Resources (beyond our class right now)

- Use a tuning parameter for our penalty
  - Basically, we need to figure out what the best penalty is for our model
  - We use the training set to determine the best penality
  - Videos that includes tuning parameter with LASSO
    - TidyTuesday video on LASSO with interactions
- More info on `glmnet` function
- Performing cross-validation
  - Under Cross validation within Data Science in a Box
- For complete video of machine learning with **LASSO**, **cross-validation**, and **tuning parameters**
  - See "Unit 5 - Deck 4: Machine learning" on this Data Science in a Box page
    - Video goes through an example with more complicated data, but can be followed with our work!

Summary

- Revisited model selection techniques and discussed how a binary outcome can be treated differently than a continuous outcome
- Discussed association vs prediction modeling
- Discussed classification: a type of machine learning!
- Introduced penalized regression as a classification method
- Performed penalized regression (specifically LASSO) to select a prediction model
- Process presented today has major flaws
  - We did not tune our parameter
  - We did not perform cross validation

For your Lab 4

- You can use purposeful selection, like we did last quarter
  - If you want to focus on **association** modeling!
  - But you will need to include at least one interaction!!
  - A good way to practice this again if you struggled with it previously

- You can try out LASSO regression
  - If you want to focus on **prediction** modeling!
  - And if you want to stretch your R coding skills