ST 558 HW 5

Alise Miller

Task 1: Conceptual Quesitons

1. The purposes of cross-validation when fitting a random forest model is to help find the best fit. So find the model that has the best metrics and that doesn't over fit for best generalization.
2. Bagged tree algorithm has many steps. The steps include having your original sample. From there getting the bootstrap samples. For non parametric we treat the sample as population, the resampling is done with replacement and we can get the same observation multiple times. After all of the trees are trained we get the bootstrap statistics.
3. General linear model is y= B_0 + B_1X_1 + B_2X_2 +B_3X_3.... This means a model that uses one or more predictors to predict the response variable. When I think of GLM is the the first model that comes to mind when making a model.
4. Adding an interaction term to MLR helps control the effect of one predictor to another. Predictors can change based on the level of other predictors.
5. We split our data into training and test sets so that we can have an accurate model and so that the data isn't over fit. The training set helps to make the model, while the testing evaulate how well the model will work for future data.

Task 2: Data Prep

packages and data

```r
suppressWarnings(library(tidyverse))
suppressWarnings(library(tidymodels))
suppressWarnings(library(caret))
suppressWarnings(library(yardstick))

heart_data <- read_csv("https://www4.stat.ncsu.edu/~online/datasets/heart.csv")
```

1. Run and report summary

```r
summary(heart_data)
```

```
      Age            Sex            ChestPainType         RestingBP    
 Min.   :28.00   Length:918         Length:918         Min.   :  0.0  
 1st Qu.:47.00   Class :character   Class :character   1st Qu.:120.0  
 Median :54.00   Mode  :character   Mode  :character   Median :130.0  
 Mean   :53.51                                         Mean   :132.4  
 3rd Qu.:60.00                                         3rd Qu.:140.0  
 Max.   :77.00                                         Max.   :200.0  
  Cholesterol      FastingBS       RestingECG            MaxHR      
 Min.   :  0.0   Min.   :0.0000   Length:918         Min.   : 60.0  
 1st Qu.:173.2   1st Qu.:0.0000   Class :character   1st Qu.:120.0  
 Median :223.0   Median :0.0000   Mode  :character   Median :138.0  
 Mean   :198.8   Mean   :0.2331                      Mean   :136.8  
 3rd Qu.:267.0   3rd Qu.:0.0000                      3rd Qu.:156.0  
 Max.   :603.0   Max.   :1.0000                      Max.   :202.0  
 ExerciseAngina        Oldpeak          ST_Slope          HeartDisease   
 Length:918         Min.   :-2.6000   Length:918         Min.   :0.0000  
 Class :character   1st Qu.: 0.0000   Class :character   1st Qu.:0.0000  
 Mode  :character   Median : 0.6000   Mode  :character   Median :1.0000  
                    Mean   : 0.8874                      Mean   :0.5534  
                    3rd Qu.: 1.5000                      3rd Qu.:1.0000  
                    Max.   : 6.2000                      Max.   :1.0000  
```

```r
str(heart_data)
```

```
spc_tbl_ [918 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ Age           : num [1:918] 40 49 37 48 54 39 45 54 37 48 ...
 $ Sex           : chr [1:918] "M" "F" "M" "F" ...
 $ ChestPainType : chr [1:918] "ATA" "NAP" "ATA" "ASY" ...
 $ RestingBP     : num [1:918] 140 160 130 138 150 120 130 110 140 120 ...
 $ Cholesterol   : num [1:918] 289 180 283 214 195 339 237 208 207 284 ...
 $ FastingBS     : num [1:918] 0 0 0 0 0 0 0 0 0 0 ...
 $ RestingECG    : chr [1:918] "Normal" "Normal" "ST" "Normal" ...
 $ MaxHR         : num [1:918] 172 156 98 108 122 170 170 142 130 120 ...
 $ ExerciseAngina: chr [1:918] "N" "N" "N" "Y" ...
 $ Oldpeak       : num [1:918] 0 1 0 1.5 0 0 0 0 1.5 0 ...
 $ ST_Slope      : chr [1:918] "Up" "Flat" "Up" "Flat" ...
 $ HeartDisease  : num [1:918] 0 1 0 1 0 0 0 0 1 0 ...
 - attr(*, "spec")=
  .. cols(
  ..   Age = col_double(),
  ..   Sex = col_character(),
  ..   ChestPainType = col_character(),
  ..   RestingBP = col_double(),
  ..   Cholesterol = col_double(),
  ..   FastingBS = col_double(),
  ..   RestingECG = col_character(),
  ..   MaxHR = col_double(),
  ..   ExerciseAngina = col_character(),
  ..   Oldpeak = col_double(),
  ..   ST_Slope = col_character(),
  ..   HeartDisease = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

1. a. In R Heart Disease is "num" so it is understood as quantitative. 1.b. This doesn't make since because these are it's yes or no and it does not make sense to average yes or no. Maybe count or percent of Yes versus No's makes better sense.

2. Change Heart Disease

```r
new_heart <- heart_data |>
  mutate(heartdisease_yes_no =
           factor(HeartDisease,
                  levels = c(0,1),
                  labels = c("No", "Yes"))
  ) |>
  select(-HeartDisease, -ST_Slope)
glimpse(new_heart) #just to check
```

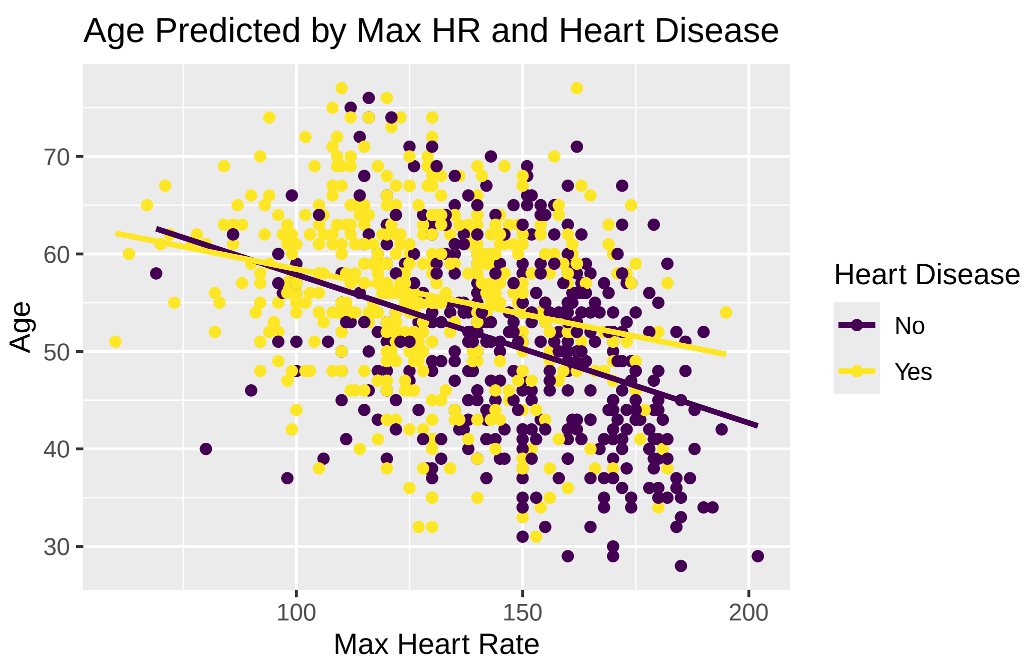
```
Rows: 918
Columns: 11
$ Age                 <dbl> 40, 49, 37, 48, 54, 39, 45, 54, 37, 48, 37, 58, 39~
$ Sex                 <chr> "M", "F", "M", "F", "M", "M", "F", "M", "M", "F", ~
$ ChestPainType       <chr> "ATA", "NAP", "ATA", "ASY", "NAP", "NAP", "ATA", "~
$ RestingBP           <dbl> 140, 160, 130, 138, 150, 120, 130, 110, 140, 120, ~
$ Cholesterol         <dbl> 289, 180, 283, 214, 195, 339, 237, 208, 207, 284, ~
$ FastingBS           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ RestingECG          <chr> "Normal", "Normal", "ST", "Normal", "Normal", "Nor~
$ MaxHR               <dbl> 172, 156, 98, 108, 122, 170, 170, 142, 130, 120, 1~
$ ExerciseAngina      <chr> "N", "N", "N", "Y", "N", "N", "N", "N", "Y", "N", ~
$ Oldpeak             <dbl> 0.0, 1.0, 0.0, 1.5, 0.0, 0.0, 0.0, 0.0, 1.5, 0.0, ~
$ heartdisease_yes_no <fct> No, Yes, No, Yes, No, No, No, No, Yes, No, No, Yes~
```

Task 3 EDA

1. Plot

```r
ggplot(new_heart,
       aes(x= MaxHR, y = Age, color = heartdisease_yes_no)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(
    x= "Max Heart Rate",
    y= "Age",
  title = "Age Predicted by Max HR and Heart Disease",
  color = "Heart Disease") +
  scale_color_viridis_d()
```

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



2.Interpret Plot

It seems like both lines have a downward slope, but they intersect so they are not the same slope. From the scatter plot it seems like people that have heart disease are older and have a lower max heart heart. While people with no heart disease are younger and have a higher max heart rate. The line for people with and without heart disease seem to have a similar y-intercept with the no heart disease being slightly higher. With all of this I think that it is best we use an interaction model.

Task 4 Testing and Training

```
set.seed(101)
new_heart_split <- initial_split(new_heart, prop = 0.8)
new_heart_train <-training (new_heart_split)
new_heart_test <-testing (new_heart_split)
```

Task 5: OLS and LASSO

1. Interaction Model

```r
ols_mlr <- lm(formula = Age ~ MaxHR + heartdisease_yes_no
              + MaxHR*heartdisease_yes_no,
              data = new_heart_train )
summary(ols_mlr)
```

```

Call:
lm(formula = Age ~ MaxHR + heartdisease_yes_no + MaxHR * heartdisease_yes_no,
    data = new_heart_train)

Residuals:
     Min       1Q   Median       3Q      Max 
-22.7703  -5.7966   0.4516   5.7772  20.6378 

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)    
(Intercept)                  75.58896    3.07510  24.581  < 2e-16 ***
MaxHR                        -0.16992    0.02064  -8.233 8.43e-16 ***
heartdisease_yes_noYes       -8.58502    3.83433  -2.239  0.02546 *  
MaxHR:heartdisease_yes_noYes  0.08343    0.02716   3.072  0.00221 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.478 on 730 degrees of freedom
Multiple R-squared:  0.1839,	Adjusted R-squared:  0.1806 
F-statistic: 54.84 on 3 and 730 DF,  p-value: < 2.2e-16
```

2. Predict and RMSE

```r
pred1 <- predict(ols_mlr, newdata = new_heart_test)
results1 <- data.frame(.pred = pred1, Age = new_heart_test$Age)

OLS_rmse <- rmse(results1, truth = Age, estimate = .pred)
```

3. LASSO Recipe

```r
new_heart_cv_folds <- vfold_cv(new_heart_train, 10)
LASSO_recipe <-
  recipe( Age ~ MaxHR + heartdisease_yes_no, data = new_heart_train) |>
  step_dummy(all_nominal_predictors()) |>
  step_normalize(all_numeric_predictors()) |>
  step_interact(terms = ~ MaxHR:starts_with("heartdisease_yes_no"))
LASSO_recipe
```

```
-- Recipe ----------------------------------------------------------------------



-- Inputs

Number of variables by role

outcome:   1
predictor: 2



-- Operations

* Dummy variables from: all_nominal_predictors()

* Centering and scaling for: all_numeric_predictors()

* Interactions with: MaxHR:starts_with("heartdisease_yes_no")
```

4. LASSO Spec and Grid

```r
LASSO_spec <- linear_reg(penalty = tune(), mixture = 1) |>
  set_engine("glmnet")

LASSO_wkf <-workflow() |>
  add_recipe(LASSO_recipe) |>
  add_model(LASSO_spec)
LASSO_wkf
```

```
== Workflow ====================================================================
Preprocessor: Recipe
Model: linear_reg()

-- Preprocessor ----------------------------------------------------------------
3 Recipe Steps

* step_dummy()
* step_normalize()
* step_interact()

-- Model -----------------------------------------------------------------------
Linear Regression Model Specification (regression)

Main Arguments:
  penalty = tune()
  mixture = 1

Computational engine: glmnet
```

```r
LASSO_grid <-LASSO_wkf |>
tune_grid(resamples = new_heart_cv_folds,
          grid = grid_regular(penalty(), levels = 200),
     metrics = metric_set(rmse))
```

```
Warning: package 'glmnet' was built under R version 4.5.1
```

```r
LASSO_grid
```

```
# Tuning results
# 10-fold cross-validation
# A tibble: 10 x 4
```

```
   splits           id     .metrics           .notes
   <list>           <chr>  <list>             <list>
 1 <split [660/74]> Fold01 <tibble [200 x 5]> <tibble [0 x 3]>
 2 <split [660/74]> Fold02 <tibble [200 x 5]> <tibble [0 x 3]>
 3 <split [660/74]> Fold03 <tibble [200 x 5]> <tibble [0 x 3]>
 4 <split [660/74]> Fold04 <tibble [200 x 5]> <tibble [0 x 3]>
 5 <split [661/73]> Fold05 <tibble [200 x 5]> <tibble [0 x 3]>
 6 <split [661/73]> Fold06 <tibble [200 x 5]> <tibble [0 x 3]>
 7 <split [661/73]> Fold07 <tibble [200 x 5]> <tibble [0 x 3]>
 8 <split [661/73]> Fold08 <tibble [200 x 5]> <tibble [0 x 3]>
 9 <split [661/73]> Fold09 <tibble [200 x 5]> <tibble [0 x 3]>
10 <split [661/73]> Fold10 <tibble [200 x 5]> <tibble [0 x 3]>
```

```
LASSO_grid |>
  collect_metrics() |>
  filter(.metric == "rmse")
```

```
# A tibble: 200 x 7
    penalty .metric .estimator  mean     n std_err .config
      <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
 1 1    e-10 rmse    standard    8.50    10   0.162 Preprocessor1_Model001
 2 1.12e-10 rmse    standard    8.50    10   0.162 Preprocessor1_Model002
 3 1.26e-10 rmse    standard    8.50    10   0.162 Preprocessor1_Model003
 4 1.41e-10 rmse    standard    8.50    10   0.162 Preprocessor1_Model004
 5 1.59e-10 rmse    standard    8.50    10   0.162 Preprocessor1_Model005
 6 1.78e-10 rmse    standard    8.50    10   0.162 Preprocessor1_Model006
 7 2.00e-10 rmse    standard    8.50    10   0.162 Preprocessor1_Model007
 8 2.25e-10 rmse    standard    8.50    10   0.162 Preprocessor1_Model008
 9 2.52e-10 rmse    standard    8.50    10   0.162 Preprocessor1_Model009
10 2.83e-10 rmse    standard    8.50    10   0.162 Preprocessor1_Model010
# i 190 more rows
```

```
lowest_rmse <- LASSO_grid |>
  select_best(metric = "rmse")

lowest_rmse
```

```
# A tibble: 1 x 2
  penalty .config
    <dbl> <chr>
1  0.0174 Preprocessor1_Model165
```

```r
LASSO_wkf |>
  finalize_workflow(lowest_rmse)
```

```
== Workflow ====================================================================
Preprocessor: Recipe
Model: linear_reg()

-- Preprocessor ----------------------------------------------------------------
3 Recipe Steps

* step_dummy()
* step_normalize()
* step_interact()

-- Model -----------------------------------------------------------------------
Linear Regression Model Specification (regression)

Main Arguments:
  penalty = 0.0174263338600965
  mixture = 1

Computational engine: glmnet
```

```r
LASSO_final <- LASSO_wkf |>
  finalize_workflow(lowest_rmse) |>
  fit(new_heart_train)

LASSO_rmse_val <- LASSO_final |>
  predict(new_data = new_heart_test) |>
  pull(.pred) |>
  yardstick::rmse_vec(truth = new_heart_test$Age)

LASSO_rmse_val
```

```
[1] 9.095981
```

```r
tidy(LASSO_final)
```

```
# A tibble: 4 x 3
  term                            estimate penalty
  <chr>                                <dbl>   <dbl>
1 (Intercept)                          54.0   0.0174
2 MaxHR                                -3.08  0.0174
3 heartdisease_yes_no_Yes               1.36  0.0174
4 MaxHR_x_heartdisease_yes_no_Yes       1.03  0.0174
```

5. RMSE

I think that the RMSE calculation to be different, because I think the LASSO would be more accurate. IT has more "training" than the OLS.

6. RMSE OLS vs LASSO

```
OLS_rmse
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard        9.10
```

```
LASSO_rmse_val
```

```
[1] 9.095981
```

The RMSE for OLS is 9.10 while the RMSE for the LASSO data is 9.096. Although they are LASSO is slightly better, they are roughly the same if you keep 2 decimal places.

7. Why are they similar?

I think they are similar because the penalty of the LASSO was very small. So that might mean that there wasn't anything to shrink.

Task 6: Logistic Regression

```r
LR1_recipe <- recipe(heartdisease_yes_no ~ Sex + Age, data = new_heart_train) |>
  step_normalize(Age) |>
  step_dummy(Sex)

LR2_recipe <- recipe(heartdisease_yes_no ~ Sex + ChestPainType + Age, data = new_heart_train
  step_normalize(all_numeric_predictors()) |>
  step_dummy(all_nominal_predictors())
LR_spec <- logistic_reg() |>
  set_engine("glm")
LR1_wf <- workflow() |>
  add_recipe(LR1_recipe) |>
  add_model(LR_spec)
LR2_wf <- workflow() |>
  add_recipe(LR2_recipe) |>
  add_model(LR_spec)
LR1_fit <- fit(LR1_wf, data = new_heart_test)
LR2_fit <- fit(LR2_wf, data = new_heart_test)

LR1_fit
```

```
== Workflow [trained] ==========================================================
Preprocessor: Recipe
Model: logistic_reg()

-- Preprocessor ----------------------------------------------------------------
2 Recipe Steps

* step_normalize()
* step_dummy()

-- Model -----------------------------------------------------------------------

Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)

Coefficients:
(Intercept)          Age        Sex_M
    -1.3794       0.5833       1.7050

Degrees of Freedom: 183 Total (i.e. Null);  181 Residual
Null Deviance:      255
Residual Deviance: 222.4    AIC: 228.4
```

```
LR2_fit
```

```
== Workflow [trained] ==========================================================
Preprocessor: Recipe
Model: logistic_reg()

-- Preprocessor ----------------------------------------------------------------
2 Recipe Steps

* step_normalize()
* step_dummy()

-- Model -----------------------------------------------------------------------

Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)

Coefficients:
      (Intercept)                Age              Sex_M  ChestPainType_ATA
         -0.06065            0.57006            1.36377           -3.47513
ChestPainType_NAP   ChestPainType_TA
         -1.63782           -1.07272

Degrees of Freedom: 183 Total (i.e. Null);  178 Residual
Null Deviance:      255
Residual Deviance: 172.7    AIC: 184.7
```

With resampling

```
repeatnew_heart_cv_folds <- vfold_cv(new_heart_train, v= 10, repeats = 5 )
LR1_resampling <- LR1_wf |>
  fit_resamples(repeatnew_heart_cv_folds, metrics = metric_set(accuracy, mn_log_loss))
LR2_resampling <- LR2_wf |>
  fit_resamples(repeatnew_heart_cv_folds, metrics = metric_set(accuracy, mn_log_loss))

LR1_resampling
```

```
# Resampling results
# 10-fold cross-validation repeated 5 times
# A tibble: 50 x 5
   splits           id      id2    .metrics         .notes
   <list>           <chr>   <chr>  <list>           <list>
```

```
 1 <split [660/74]> Repeat1 Fold01 <tibble [2 x 4]> <tibble [0 x 3]>
 2 <split [660/74]> Repeat1 Fold02 <tibble [2 x 4]> <tibble [0 x 3]>
 3 <split [660/74]> Repeat1 Fold03 <tibble [2 x 4]> <tibble [0 x 3]>
 4 <split [660/74]> Repeat1 Fold04 <tibble [2 x 4]> <tibble [0 x 3]>
 5 <split [661/73]> Repeat1 Fold05 <tibble [2 x 4]> <tibble [0 x 3]>
 6 <split [661/73]> Repeat1 Fold06 <tibble [2 x 4]> <tibble [0 x 3]>
 7 <split [661/73]> Repeat1 Fold07 <tibble [2 x 4]> <tibble [0 x 3]>
 8 <split [661/73]> Repeat1 Fold08 <tibble [2 x 4]> <tibble [0 x 3]>
 9 <split [661/73]> Repeat1 Fold09 <tibble [2 x 4]> <tibble [0 x 3]>
10 <split [661/73]> Repeat1 Fold10 <tibble [2 x 4]> <tibble [0 x 3]>
# i 40 more rows
```

```
LR2_resampling
```

```
# Resampling results
# 10-fold cross-validation repeated 5 times
# A tibble: 50 x 5
   splits           id      id2    .metrics         .notes
   <list>           <chr>   <chr>  <list>           <list>
 1 <split [660/74]> Repeat1 Fold01 <tibble [2 x 4]> <tibble [0 x 3]>
 2 <split [660/74]> Repeat1 Fold02 <tibble [2 x 4]> <tibble [0 x 3]>
 3 <split [660/74]> Repeat1 Fold03 <tibble [2 x 4]> <tibble [0 x 3]>
 4 <split [660/74]> Repeat1 Fold04 <tibble [2 x 4]> <tibble [0 x 3]>
 5 <split [661/73]> Repeat1 Fold05 <tibble [2 x 4]> <tibble [0 x 3]>
 6 <split [661/73]> Repeat1 Fold06 <tibble [2 x 4]> <tibble [0 x 3]>
 7 <split [661/73]> Repeat1 Fold07 <tibble [2 x 4]> <tibble [0 x 3]>
 8 <split [661/73]> Repeat1 Fold08 <tibble [2 x 4]> <tibble [0 x 3]>
 9 <split [661/73]> Repeat1 Fold09 <tibble [2 x 4]> <tibble [0 x 3]>
10 <split [661/73]> Repeat1 Fold10 <tibble [2 x 4]> <tibble [0 x 3]>
# i 40 more rows
```

```
collect_metrics(LR1_resampling)
```

```
# A tibble: 2 x 6
  .metric     .estimator  mean     n std_err .config
  <chr>       <chr>      <dbl> <int>   <dbl> <chr>
1 accuracy    binary     0.675    50 0.00647 Preprocessor1_Model1
2 mn_log_loss binary     0.601    50 0.00571 Preprocessor1_Model1
```

```r
collect_metrics(LR2_resampling)
```

```
# A tibble: 2 x 6
  .metric     .estimator  mean     n std_err .config
  <chr>       <chr>      <dbl> <int>   <dbl> <chr>
1 accuracy    binary     0.777    50 0.00660 Preprocessor1_Model1
2 mn_log_loss binary     0.488    50 0.00959 Preprocessor1_Model1
```

```r
model1 <- collect_metrics(LR1_resampling) |>
  mutate(Model = "Model1")
model2 <- collect_metrics(LR2_resampling) |>
  mutate(Model = "Model2")
combined_metric <- rbind(model1, model2)

combined_metric
```

```
# A tibble: 4 x 7
  .metric     .estimator  mean     n std_err .config              Model
  <chr>       <chr>      <dbl> <int>   <dbl> <chr>                <chr>
1 accuracy    binary     0.675    50 0.00647 Preprocessor1_Model1 Model1
2 mn_log_loss binary     0.601    50 0.00571 Preprocessor1_Model1 Model1
3 accuracy    binary     0.777    50 0.00660 Preprocessor1_Model1 Model2
4 mn_log_loss binary     0.488    50 0.00959 Preprocessor1_Model1 Model2
```

I think my best performing model is Model 2, because it has the higher accuracy (.773) and the lowest log loss (.489) means.

2 Confusion Matrix

```r
LR2_fit <- fit(LR2_wf, data = new_heart_train)
prediction <- predict(LR2_fit, new_data = new_heart_test) |>
  pull() |>
  factor(levels = c("No", "Yes"))
confusionMatrix(
  data = prediction,
  reference = new_heart_test$heartdisease_yes_no
)
```

```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  73  16
       Yes 21  74

               Accuracy : 0.7989
                 95% CI : (0.7336, 0.8543)
    No Information Rate : 0.5109
    P-Value [Acc > NIR] : 5.393e-16

                  Kappa : 0.5981

 Mcnemar's Test P-Value : 0.5108

            Sensitivity : 0.7766
            Specificity : 0.8222
         Pos Pred Value : 0.8202
         Neg Pred Value : 0.7789
             Prevalence : 0.5109
         Detection Rate : 0.3967
   Detection Prevalence : 0.4837
      Balanced Accuracy : 0.7994

       'Positive' Class : No
```

The accuracy is about 80%.

3 Sensitivity and Specificity and interpretation

Sensitivity : 0.7766
Specificity : 0.8222

Sensitivity, 77.66% is the percent of people who don't have heart disease that receive the negative result. Specificity, 82.22% is the percent of people who truly have heart disease that are correctly identified. This reminds me of false positives and false negatives in inference.