HW5 - Model Comparison

Kayla Kippes

Task 1: Conceptual Questions

- What is the purpose of using cross-validation when fitting a random forest model?

Cross-validation is used when fitting a random forest model in order to split the data into multiple folds and only train the data on some of those folds while testing it on others. This helps pick the best model by seeing how the model will do on unseen data and it will help prevent overfitting.

- Describe the bagged tree algorithm.

Create many bootstrap samples of the data and then fit trees to each of those samples of the training data. For each tree, find a prediction value and take all predictions and either average (for regression) or chose the most common (for classification) to create one final prediction.

- What is meant by a general linear model?

General linear model is a group of models that allows for the response variable to taken on different distributions so the predictors could be continuous or categorical. One example would be modeling a response variable that is binary.

- When fitting a multiple linear regression model, what does adding an interaction term do? That is, what does it allow the model to do differently as compared to when it is not included in the model?

Adding an interaction term allows for the value of one predictor be dependent on the value of another predictor. Without it, the model assumes that each predictor affects the response independently.

- Why do we split our data into a training and test set?

We split our data into a training and test set so that the model can be fit on data it has never seen before. It can learn from the training data and from there we can evaluate it's performance on the test data.

Task 2: Data Prep

Packages and Data

```r
## load the necessary packages
library(tidyverse)
library(tidymodels)
library(caret)
library(yardstick)

## load the data and save it as a tibble
heart <- read_csv('heart.csv') |>
  as.tibble()
```

Question 1

```r
## summary of the heart data set
summary(heart)
```

```
      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  
```

a. Heart Disease is currently a quantitative variable.

b. This does not make sense because it is a binary classification (0 or 1) so it should be a factor.

Question 2

```r
## fix heart disease variable type and select all but two columns
new_heart <- heart |>
  mutate(HD_Indicator = as.factor(HeartDisease)) |>
  select(-ST_Slope, -HeartDisease)
```

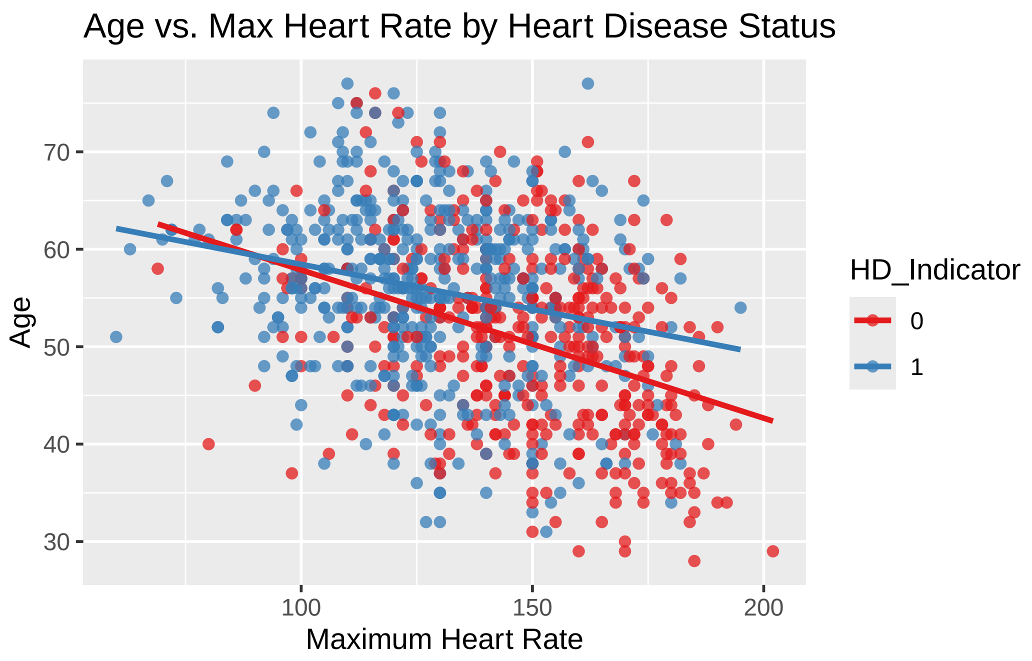
Task 3: EDA

Question 1

```r
## load ggplot
library(ggplot2)

## plot visual
ggplot(new_heart, aes(x = MaxHR, y = Age, color = HD_Indicator)) +
  geom_point(alpha = 0.75) +
  geom_smooth(method = "lm", se = FALSE) +
  scale_color_brewer(palette = "Set1") +
  labs(
  title = "Age vs. Max Heart Rate by Heart Disease Status",
  x = "Maximum Heart Rate",
  y = "Age")
```

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



Question 2

Based on the plot above, I think an interaction model is more appropriate here due to the differing slopes for each factor of the heart disease indicator.

Task 4: Testing and Training

```
## set seed
set.seed(101)

## split the data into train and test sets
heart_split <- initial_split(new_heart, prop = 0.8)
train <- training(heart_split)
test <- testing(heart_split)
```

Task 5: OLS and LASSO

Question 1

```r
## Fit an interaction model
ols_mlr <- lm(Age ~ MaxHR * HD_Indicator, data = train)

## summary of the output
summary(ols_mlr)
```

```

Call:
lm(formula = Age ~ MaxHR * HD_Indicator, data = 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 ***
HD_Indicator1       -8.58502    3.83433  -2.239  0.02546 *  
MaxHR:HD_Indicator1  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
```

Question 2

```r
#look at a few observations and predictions
test |>
  mutate(model_preds = predict(ols_mlr, newdata = test)) |>
  rmse(truth = Age, estimate = model_preds)
```

```
# A tibble: 1 x 3
```

```
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard        9.10
```

Question 3

```
## set up cv folds
cv_folds <- vfold_cv(train, 10)

## set up LASSO recipe
LASSO_recipe <- recipe(Age ~ MaxHR + HD_Indicator, data = train) |>
  step_dummy(all_nominal_predictors()) |> ## convert to dummary varaible
  step_normalize(all_predictors()) |> ## standardize to normal
  step_interact(terms = ~ MaxHR:starts_with("HD_Indicator"))
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_predictors()

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

Question 4

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

## Workflow
LASSO_workflow <- workflow() |>
  add_recipe(LASSO_recipe) |>
  add_model(LASSO_spec)

## Grid
LASSO_grid <- LASSO_workflow |>
  tune_grid(resamples = cv_folds,
            grid = grid_regular(penalty(),
                                levels = 200))

## Best model
LASSO_best <- LASSO_grid |>
  select_best(metric = "rmse")

## Finalize
final_LASSO <- LASSO_workflow |>
  finalize_workflow(LASSO_best) |>
  fit(train)

tidy(final_LASSO)
```

```
# A tibble: 4 x 3
  term                    estimate penalty
  <chr>                      <dbl>   <dbl>
1 (Intercept)                54.0   0.0174
2 MaxHR                      -3.08  0.0174
3 HD_Indicator_X1             1.36  0.0174
4 MaxHR_x_HD_Indicator_X1     1.03  0.0174
```

Question 5

I would expect the RMSE calculations to be roughly the same. This is because the penalty in the LASSO model is close to 0 so it should essentially be like an OLS model.

Question 6

```
#Calculate test RMSE
ols_rmse <- sqrt(mean((test$Age-predict(ols_mlr,newdata = test))^2))

# Get rmse from our LASSO model
LASSO_test_rmse <- LASSO_workflow |>
  finalize_workflow(LASSO_best) |>
  last_fit(heart_split) |>
  collect_metrics() |>
  filter(.metric == "rmse")

# Combine our metrics
rbind(ols_rmse, LASSO_test_rmse) |>
  mutate(Model = c("OLS", "LASSO")) |>
  select(Model, everything())
```

```
# A tibble: 2 x 5
  Model .metric          .estimator       .estimate .config
  <chr> <chr>            <chr>                <dbl> <chr>
1 OLS   9.10020630209785 9.10020630209785      9.10 9.10020630209785
2 LASSO rmse             standard              9.10 Preprocessor1_Model1
```

Question 7

Although the models may have different coefficients, the RMSE's were the same. This is because LASSO shrinks coefficients but because there was not as large of a penalty, there wasn't severe shrinkage and the number of predictors was essentially unchanged.

Task 6: Logistic Regression

Question 1

```
## Set up two different logistic regression models

## Model 1
LR1_rec <- recipe(HD_Indicator ~ Age + RestingBP, data = train) |>
  step_normalize(all_predictors())
```

```
LR2_rec <- recipe(HD_Indicator ~ Age + RestingBP + Cholesterol + MaxHR, data = train) |>
  step_normalize(all_predictors())

## spec for both models
LR_spec <- logistic_reg() |>
  set_engine("glm")

## create workflows
LR1_wkf <- workflow() |>
 add_recipe(LR1_rec) |>
 add_model(LR_spec)

LR2_wkf <- workflow() |>
 add_recipe(LR2_rec) |>
 add_model(LR_spec)

## Fit to the CV folds
LR1_fit <- LR1_wkf |>
 fit_resamples(cv_folds, metrics = metric_set(accuracy, mn_log_loss))

LR2_fit <- LR2_wkf |>
 fit_resamples(cv_folds, metrics = metric_set(accuracy, mn_log_loss))

## Combine and see which did best
rbind(LR1_fit |> collect_metrics(),
 LR2_fit |> collect_metrics()) |>
 mutate(Model = c("Model1", "Model1", "Model2", "Model2")) |>
 select(Model, everything())
```

```
# A tibble: 4 x 7
  Model  .metric     .estimator  mean     n std_err .config
  <chr>  <chr>       <chr>      <dbl> <int>   <dbl> <chr>
1 Model1 accuracy    binary     0.639    10  0.0172 Preprocessor1_Model1
2 Model1 mn_log_loss binary     0.645    10  0.0128 Preprocessor1_Model1
3 Model2 accuracy    binary     0.685    10  0.0199 Preprocessor1_Model1
4 Model2 mn_log_loss binary     0.585    10  0.0165 Preprocessor1_Model1
```

Model 2 (using Age, Resting BP, Cholesterol, and MaxHR) is the best one due to higher accuracy and lower log loss - although it is not too much better.

Question 2

```r
## Fit the best model
final_LR_fit <- LR2_wkf |>
  fit(train)

conf_mat(test |> mutate(estimate = final_LR_fit |> predict(test) |> pull()),
 HD_Indicator,
 estimate)
```

```
          Truth
Prediction  0  1
         0 70 21
         1 24 69
```

Question 3

Sensitivity: Of all the people who have heart disease, the model correctly identified 69 out of the 184 (true positive rate).

Specificity: Of all the people who DO NOT have heart disease, the model correctly identified 70 out of the 184 (true negative rate).