Homework 5 Julia Fish

Task 1: Conceptual Questions

- Question 1: **What is the purpose of using cross-validation when fitting a random forest model?**
  Cross-validation helps estimate a model's performance with "new" data. When fitting a random forest model, cross-validation ensures that the model is not overfitting by evaluating its performance across multiple data splits (i.e. leaving one fold out at a time). Because of this, it gives a more reliable estimate of prediction error than a single train-test split.

- Question 2: **Describe the bagged tree algorithm.**
  The baged tree algorithm involves generating multiple training sets using bootstrapping techniques. A decision tree is trained on each of the bootstrap resamples, and predictions are averaged over. This reduces variance and increases model stability compared to fitting a single tree model.

- Question 3: **What is meant by a general linear model?**
  A general linear model is a linear model that can have different forms of response variable than just all real numbers. For example, binomial, poisson, gamma, etc. can be used in general linear models to help modify the response range as well as oher logistical factors in the model fit.

- Question 4: **When fitting a multiple linear regression model, what does adding an interaction term do?**
  Adding an interaction term to a MLR model allows the effect of one variable on the response to depend on the level of another variable. That is, the variables are allowed to communicate with one another to predict a response value instead of only having just additive effects.

- Question 5: **Why do we split our data into a training and test set?**
  Using a training and test set allows us to see model performance on "unseen" data with known response values. That way we can get a better idea of whether or not our model performs well or has just been overfit to the data it was given to train the model.

Task 2: Data Prep

Packages and Data

First, we will load in the packages we need for this task. We will also read in our `heart` data set as a tibble.

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

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

Question 1

Now, we will run and discuss the `summary()` of this tibble.

```r
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  
```

Above, we see the full summary of this data set. The `HeartDisease` variable seems to be a quantitative variable, which does not make sense in this case since it is supposed to represent the presence or absence of heart disease. That should be categorical.

Question 2

Now, we will create a categorical`HeartDisease` variable named `HD`. In the same pipeline, we will also remove `ST_Slope` and the original `HeartDisease` variable. Lastly, we will save this new version of the data under `new_heart`. We will do this below:

```r
new_heart <- heart |>
  mutate(HD = factor(HeartDisease)) |>
  select(-ST_Slope, -HeartDisease)
```

Task 3: EDA

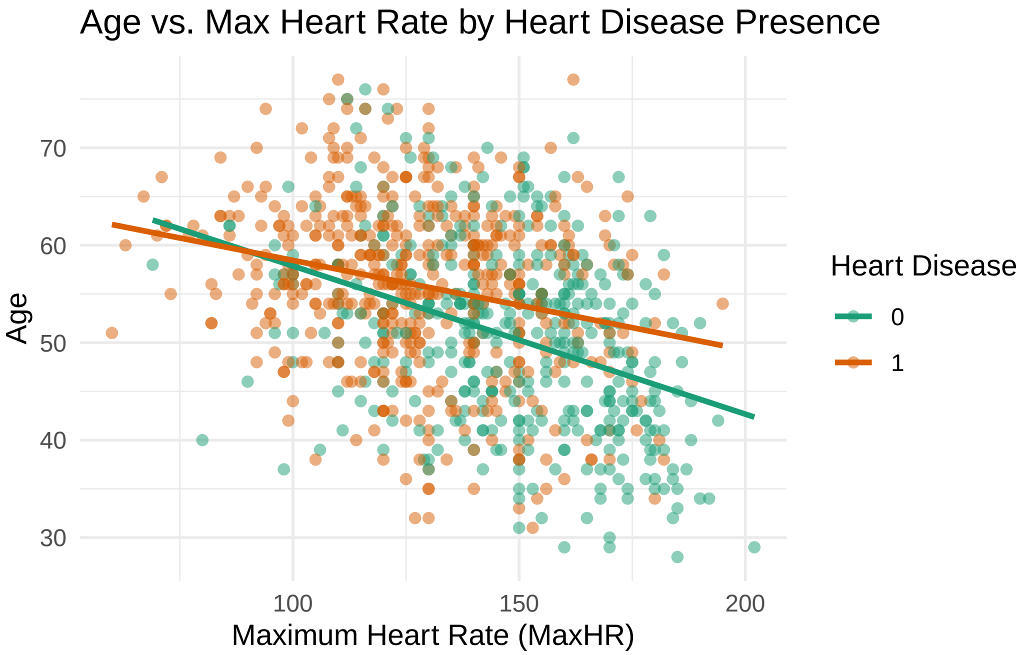
Question 1

Here, we hope to make a scatterplot to visualize age vs. a function of heart disease and their max heart rate. We will do this by making the x-axis contain max heart rate, the y-axis containing age, and the points being colored by whether or not the person has heart disease with a colorblind friendly pallet. Then, the trend lines will be put on the plot as well.

```r
library(ggplot2)
library(scales)

ggplot(new_heart, aes(x = MaxHR, y = Age, color = HD)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = FALSE) +
  scale_color_brewer(palette = "Dark2") +
  labs(
    title = "Age vs. Max Heart Rate by Heart Disease Presence",
    x = "Maximum Heart Rate (MaxHR)",
    y = "Age",
    color = "Heart Disease"
  ) +
  theme_minimal()
```

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



Question 2

Based on this plot, I think an interaction plot would be better than an additive model. That is because the lines are not parallel (or almost parallel). That means that the slope changes with the heart disease status, which points to an interaction model being a more appropriate fit.

Task 4: Testing and Training

Now, we will split our `new_heart` into an 80-20 spli for training and testing. This will be done below (with a seed of 101 set):

```r
set.seed(101)

heart_split <- initial_split(new_heart, prop = 0.8)

train <- training(heart_split)
test <- testing(heart_split)
```

Task 5: OLS and LASSO

Question 1

First, we will fit an OLS interaction model named `ols_mlr` as described in the homework. We will then report a summary.

```r
ols_mlr <- lm(Age ~ MaxHR * HD, data = train)
summary(ols_mlr)
```

```

Call:
lm(formula = Age ~ MaxHR * HD, 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 ***
HD1         -8.58502    3.83433  -2.239  0.02546 *  
MaxHR:HD1    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
```

From this summary, we see that the overall model is significant with an F value of 54.84 and an associated p-value of 2.2e-16. In addition, at the 0.05 significance level, all individual predictors are significant as well.

Question 2

Now, to get a better evaluation of the model's performance, we will predict on the test set and calculate the RMSE using this model below:

```r
preds <- predict(ols_mlr, newdata = test)


results <- test %>%
  mutate(pred = preds)

ols_rmse <- rmse(results, truth = Age, estimate = pred)
ols_rmse
```

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

The RMSE using this model and the test set is 9.100206.

Question 3

Now, we are going to fit a LASSO model with cross validation to compare to the OLS model fit above. We will start with our recipe as guided in the file:

```r
LASSO_recipe <- recipe(Age ~ MaxHR + HD, data = train) |>
  step_dummy(all_nominal_predictors()) |>
  step_normalize(all_numeric_predictors()) |>
  step_interact(~ MaxHR:starts_with("HD_"))

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("HD_")
```

Question 4

Now that out recipe is made, we will set up our `spec` and grid. After that, we will use those to pick a final model and report using `tidy()`. We will do that below:

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

set.seed(101)
heart_cv_folds <- vfold_cv(train, 10)


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

LASSO_grid <- LASSO_wkf |>
  tune_grid(resamples = heart_cv_folds,
            grid = grid_regular(penalty(), levels = 200))

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    8.47    10   0.124 Preprocessor1_Model001
 2 1.12e-10 rmse    standard    8.47    10   0.124 Preprocessor1_Model002
 3 1.26e-10 rmse    standard    8.47    10   0.124 Preprocessor1_Model003
 4 1.41e-10 rmse    standard    8.47    10   0.124 Preprocessor1_Model004
 5 1.59e-10 rmse    standard    8.47    10   0.124 Preprocessor1_Model005
 6 1.78e-10 rmse    standard    8.47    10   0.124 Preprocessor1_Model006
 7 2.00e-10 rmse    standard    8.47    10   0.124 Preprocessor1_Model007
 8 2.25e-10 rmse    standard    8.47    10   0.124 Preprocessor1_Model008
 9 2.52e-10 rmse    standard    8.47    10   0.124 Preprocessor1_Model009
10 2.83e-10 rmse    standard    8.47    10   0.124 Preprocessor1_Model010
# i 190 more rows
```

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

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 = 1e-10
  mixture = 1

Computational engine: glmnet
```

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

tidy(LASSO_final)
```

```
# A tibble: 4 x 3
  term          estimate      penalty
  <chr>            <dbl>        <dbl>
1 (Intercept)      54.0  0.0000000001
2 MaxHR            -3.08 0.0000000001
3 HD_X1             1.36 0.0000000001
4 MaxHR_x_HD_X1     1.03 0.0000000001
```

As seen above, we selected the final model.

Question 5

Without looking at the RMSE values, I would expect them to be lower for the LASSO model due to shrinkage of the coefficients to 0. It can be seen in the estimates column above that the coefficient estimates are much closer to 0 that for the OLS model, which leads me to believe that the variance will be lower and the RMSE will be lower as a result.

Question 6

Now, we will calculate the RMSE for the LASSO model and compare it to the OLS model, ultimately looking for very similar values.

```
preds2 <- predict(LASSO_final, new_data = test)


LASSO_results <- test %>%
  mutate(pred = preds2$.pred)

LASSO_rmse <- rmse(LASSO_results, truth = Age, estimate = pred)
LASSO_rmse
```

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

This value is very similar to the RMSE value obtained for the OLS model (9.100206).

Question 7

The RMSE calculations ended up being roughly the same even though the coefficients are different since the shrinkage is targeted toward taking out unnecessary information in the model. In addition, the shrinkage affects ALL predictor coefficients in the model, which means that the prediction can stay fairly precise even if all of the coefficients differ. Said differently, the variance decreases with the LASSO model, but the bias increases, so that trade off makes the predictions still able to be similar to the OLS model.