Homework 5

Task 1: Conceptual Questions

- What is the purpose of using cross-validation when fitting a random forest model?
  - Cross-validation when fitting random forest models is not always necessary, but it is used to tune the random forest model by selecting the number of predictors that minimizes the log-loss metric.
- Describe the bagged tree algorithm.
  - The bagged tree algorithm is as follows:

1. Bootstrap sampling from an original data set
2. For each bootstrap sample, fit a tree model
3. Average the predictions for each tree model for regression or use majority vote across all the trees for classification.

- What is meant by a general linear model?
  - A general linear model is a statistical model used for prediction and inference that contains coefficients that are only linear by nature.
- 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?
  - An interaction term allows two predictors to be dependent on each other while a model without an interaction term assumes that the predictors are independent of each other.
- Why do we split our data into a training and test set?

– Data is split into a training and test set so that the model can be "trained" on one set of data and "tested" on a set that is completely independent from that the data that the model was trained on. If the data was not split, then the test would not be indicative of how well the model is able to make predictions.

Task 2: Data Prep

Packages and Data

```
library(tidyverse)
library(tidymodels)
library(caret)
library(yardstick)
library(ggplot2)
library(viridis)

data <- read_csv("heart.csv")
```

1. Run and report summary() on your data set. Then, answer the following questions:

```
summary(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
```

a. HeartDisease is categorized as a numeric variable

b. This does not make sense because the HeartDisease is a binary variable which only takes on 0 and 1 which is moreso categorical. This being said, it would not make sense to do any numerical summaries on this variable.

2. Change HeartDisease to be the appropriate data type, and name it something different. In the same tidyverse pipeline, remove the ST_Slope variable and the original HeartDisease variable. Save your new data set as new_heart. We will use this new data set for the remainder of the assignment.

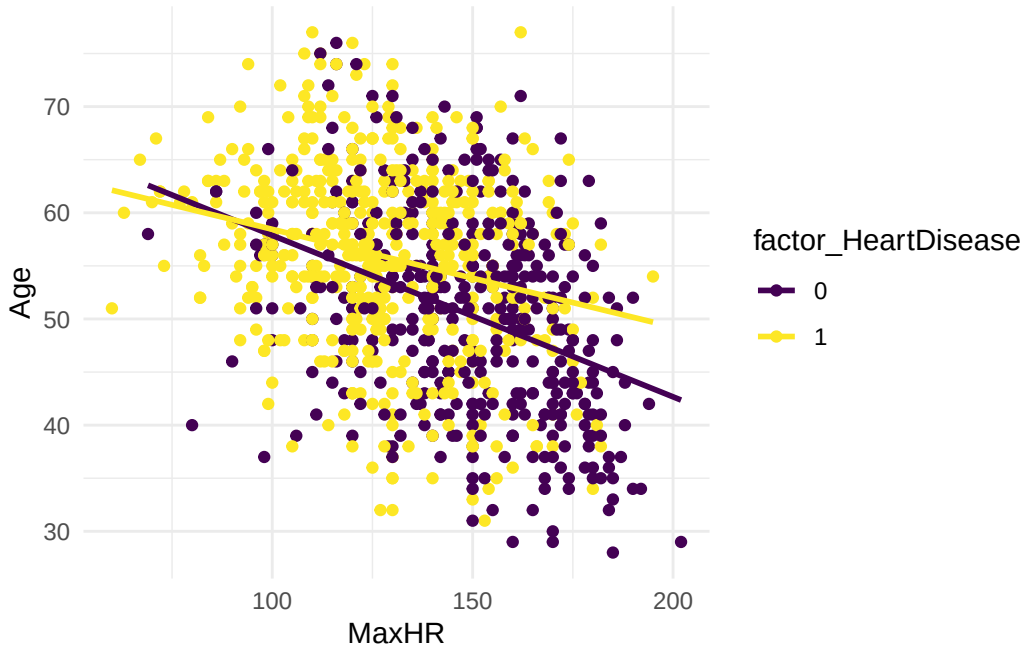
```
new_heart <- data |>
  mutate(factor_HeartDisease = factor(HeartDisease)) |>
  select(-ST_Slope, -HeartDisease)
```

Task 3: EDA

1. We are going to model someone's age (our response variable) as a function of heart disease and their max heart rate. First, create the appropriate scatterplot to visualize this relationship. Add a line to the scatterplot that represents if someone has or does not have heart disease. Remove the standard error bars from the lines and add appropriate labels. Also, change the color pallet to be more colorblind friendly.

```
ggplot(data = new_heart, aes(x = MaxHR, y = Age, color = factor_HeartDisease)) +
  geom_point() +
  geom_smooth(method = lm, se = F) +
  theme_minimal() +
  scale_color_viridis_d()
```

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



2. Based on visual evidence, do you think an interaction model or an additive model is more appropriate? Justify your answer.

Based on the visual evidence, it would be appropriate to use an interaction model because the slopes of the lines are not parallel, thus, indicating that there is interaction between the two predictor variables.

Task 4: Testing and Training

```
heart_split <- initial_split(new_heart, prop = 0.8)
heart_train <- training(heart_split)
heart_test <- testing(heart_split)
```

Task 5: OLS and LASSO

1. Regardless of your answer in Task 3, we are going to fit an interaction model. First fit an interaction model (named ols_mlr) with age as your response, and max heart rate + heart disease as your explanatory variables using the training data set using ordinary least squares regression. Report the summary output.

```
ols_mlr <- lm(Age ~ MaxHR*factor_HeartDisease, data = new_heart)

summary(ols_mlr)
```

```
Call:
lm(formula = Age ~ MaxHR * factor_HeartDisease, data = new_heart)

Residuals:
     Min       1Q   Median       3Q      Max
-23.9600  -5.8521   0.3956   5.9879  24.2622

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)                73.08380    2.73877  26.685  < 2e-16 ***
MaxHR                      -0.15209    0.01826  -8.328 2.98e-16 ***
factor_HeartDisease1       -5.43219    3.46324  -1.569   0.1171
MaxHR:factor_HeartDisease1  0.06003    0.02450   2.450   0.0145 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.601 on 914 degrees of freedom
Multiple R-squared:  0.1712,    Adjusted R-squared:  0.1685
F-statistic: 62.95 on 3 and 914 DF,  p-value: < 2.2e-16
```

2. We are going to use RMSE to evaluate this model's predictive performance on new data. Test your model on the testing data set. Calculate the residual mean square error (RMSE) and report it below.

```
predicted_ols <- predict(ols_mlr, newdata = heart_test)
```

```
rmse <- sqrt(mean((heart_test$Age - predicted_ols)^2))
rmse
```

```
[1] 9.42098
```

3. Now, we are going to see if a model fit using LASSO has better predictive performance than with OLS.

```
LASSO_recipe <- recipe(Age ~ MaxHR + factor_HeartDisease, data = heart_train) |>
  step_dummy(factor_HeartDisease) |>
  step_normalize(MaxHR) |>
  step_interact(terms = ~ MaxHR:starts_with("factor_HeartDisease"))

 LASSO_recipe
```

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

```
-- Inputs

Number of variables by role

outcome:   1
predictor: 2
```

```
-- Operations

* Dummy variables from: factor_HeartDisease

* Centering and scaling for: MaxHR

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

4. Now, set up your appropriate spec, and grid. Next, select your final model, and report the results using the tidy() function around your model name.

```
heart_cv_folds <- vfold_cv(heart_train, 10)

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

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 = 30))

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

```
# A tibble: 1 x 2
       penalty .config
         <dbl> <chr>
1 0.0000000001 Preprocessor1_Model01
```

```
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(heart_train)

tidy(LASSO_final)
```

```
# A tibble: 4 x 3
  term                          estimate      penalty
  <chr>                            <dbl>        <dbl>
1 (Intercept)                      52.4  0.0000000001
2 MaxHR                            -3.78 0.0000000001
3 factor_HeartDisease_X1            2.59 0.0000000001
4 MaxHR_x_factor_HeartDisease_X1    1.25 0.0000000001
```

5. Without looking at the RMSE calculations, would you expect the RMSE calculations to be roughly the same or different? Justify your answer using output from your LASSO model.

I would expect the RMSE calculations to be roughly the same because the predictors from the LASSO model remained the same as the predictors for the OLS model.

6. Now compare the RMSE between your OLS and LASSO model and show that the RMSE calculations were roughly the same.

```
LASSO_rmse <- LASSO_wkf |>
  finalize_workflow(lowest_rmse) |>
  last_fit(heart_split) |>
  collect_metrics() |>
  filter(.metric == "rmse") |>
  mutate(model = "LASSO") |>
  select(-.config)
```

```
OLS_rmse <- sqrt(mean((heart_test$Age - predicted_ols)^2))

OLS_rmse_tbl <- tibble(
  .metric = "rmse",
  .estimator = "standard",
  .estimate = OLS_rmse,
  model = "OLS"
)

rbind(LASSO_rmse, OLS_rmse_tbl)
```

```
# A tibble: 2 x 4
  .metric .estimator .estimate model
  <chr>   <chr>          <dbl> <chr>
1 rmse    standard        9.43 LASSO
2 rmse    standard        9.42 OLS
```

7. Why are the RMSE calculations roughly the same if the coefficients for each model are different?

The LASSO model attempts to simplify the model by reducing the coefficients to be smaller or near zero while the OLS model uses the coefficients fully. Even though they are each using different processes to simplify the model, they both balance out the errors to come to a similar RMSE.

Task 6: Logistic Regression

1. Propose two different logistic regression models with heart disease as our response.

```
LR1_rec <- recipe(factor_HeartDisease ~ RestingBP+MaxHR, data = heart_train) |>
  step_normalize(RestingBP, MaxHR)

LR2_rec <- recipe(factor_HeartDisease ~ Sex + Age + Cholesterol + RestingBP,
                  data = heart_train) |>
  step_dummy(Sex) |>
  step_normalize(RestingBP, Age, Cholesterol)
```

```
LR_spec <- logistic_reg() |>
 set_engine("glm")

LR1_wkf <- workflow() |>
 add_recipe(LR1_rec) |>
 add_model(LR_spec)
LR2_wkf <- workflow() |>
 add_recipe(LR2_rec) |>
 add_model(LR_spec)

LR1_fit <- LR1_wkf |>
 fit_resamples(heart_cv_folds, metrics = metric_set(accuracy, mn_log_loss))

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

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     0.696    10  0.0211 Preprocessor1_Model1
2 Model1 mn_log_loss binary     0.586    10  0.0182 Preprocessor1_Model1
3 Model2 accuracy    binary     0.707    10  0.0199 Preprocessor1_Model1
4 Model2 mn_log_loss binary     0.576    10  0.0156 Preprocessor1_Model1
```

Model 2 is shown to be the best performing model as it uses more predictors while Model 1 is much simpler and may not be able to distinguish between people with and without heart disease as easily.

2. Lastly, check how well your chosen model does on the test set using the confusionMatrix() function.

```
LR_train_fit <- LR2_wkf |>
 fit(heart_train)
```

```
conf_mat(heart_train |> mutate(estimate = LR_train_fit |> predict(heart_train)
                               |> pull()),
 factor_HeartDisease,
 estimate)
```

```
          Truth
Prediction   0   1
         0 180  83
         1 130 341
```

3. Next, identify the values of sensitivity and specificity, and interpret them in the context of the problem

```
specificity <- 181/(181+145)

sensitivity <- 328/(328+80)

rbind(specificity, sensitivity)
```

```
                [,1]
specificity 0.5552147
sensitivity 0.8039216
```

This model correctly identifies about 55.6% of patients that do not have heart disease correctly and about 80% of people that do have heart disease correctly.