Homework 5 - Programming in Base R

Task 1: Conceptual Questions

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

The purpose of using CV when fitting a random forest model is so that the parameters are tested on unseen data. CV allows us to find the ideal parameter that has been tested fully on data it hasn't seen. It allows for random forests to be a better model of the overall population since regular tree models can change drastically based on adding an extra observation, but with CV is is more robust.

Describe the bagged tree algorithm.

The bagged tree algorithm is the simple version of random forests, and it involves using bootstrap sampling and doing the tree on each new random sample. This allows for a reduction in the variance of the model.

What is meant by a general linear model?

A general linear model is a model that is linear in the parameters, and includes simple linear regression, multiple linear regression, and others.

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 effect of one predictor to change depending on another predictor. It allows the model to capture interactions between parameters outside of just the main effects.

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

We split our data into training and testing set to evaluate how the model does on data that it hasn't seen yet. Since we are interested in how well our models predict, training the model on the training data and then testing it on the test data allows us to evaluate how well it predicts on unseen data.

Task 2: Data Prep

Packages and Data

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

heart_data <- as_tibble(read_csv("heart.csv"))
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
```

Heart Disease is a numeric variable in R, as we can see from the above summary. However, this is a binary variable, so this is incorrect. It needs to be a categorical variable.

Now, creating the new data

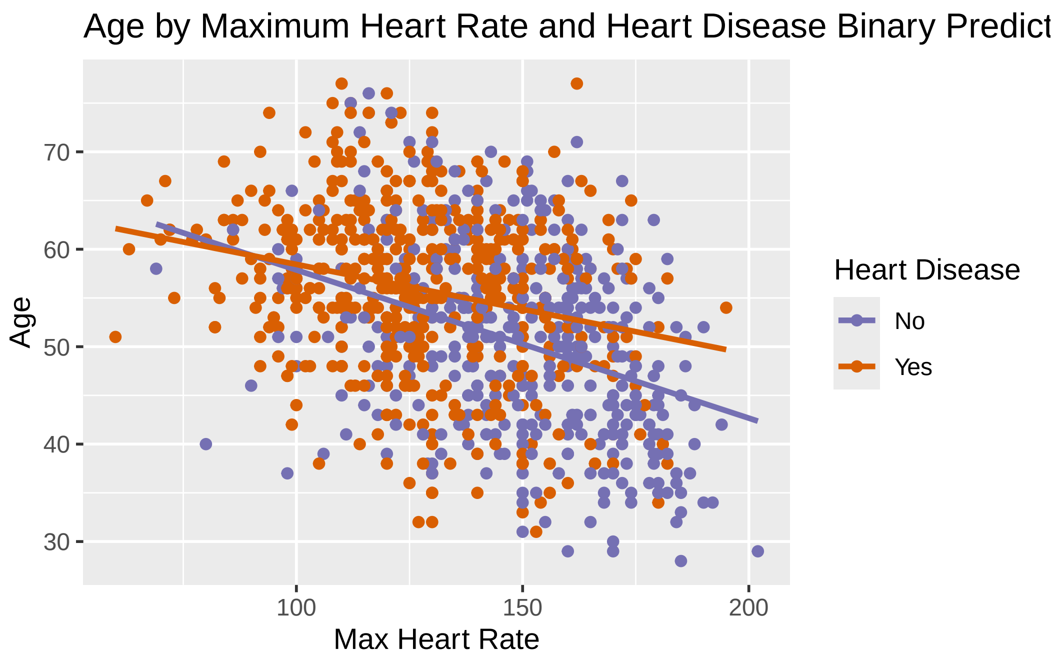
```r
new_heart <- heart_data |>
  mutate(HeartDisease_bin = factor(HeartDisease, levels = c(0,1))) |>
  select(-c(ST_Slope, HeartDisease))
```

Task 3: EDA

EDA Plot

```r
ggplot(new_heart, aes(x = MaxHR, y = Age, color = HeartDisease_bin)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Age by Maximum Heart Rate and Heart Disease Binary Predictor",
       x = "Max Heart Rate",
       y = "Age",
       color = "Heart Disease") +
  #Found colors using color brewer colorblind friendly palette
  scale_colour_manual(values = c("#7570b3", "#d95f02"), labels = c("No", "Yes"))
```

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



Based on the visual evidence, it looks like an interaction model is more appropriate because there are visible differences between the slope and intercept of the linear regression lines for Heart Disease = No versus Heart Disease = Yes.

Task 4: Testing and Training

Splitting data into training and testing

```
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

Fitting Interaction Model

```
#Interaction Model
ols_mlr <- lm(Age ~ MaxHR*HeartDisease_bin, data = train)
```

```r
#Summary
summary(ols_mlr)
```

```

Call:
lm(formula = Age ~ MaxHR * HeartDisease_bin, 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 ***
HeartDisease_bin1       -8.58502    3.83433  -2.239  0.02546 *  
MaxHR:HeartDisease_bin1  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
```

```r
#RMSE of test data
ols_rmse <- ols_mlr |>
  predict(test) |>
  rmse_vec(truth = test$Age)
ols_rmse
```

```
[1] 9.100206
```

LASSO

```r
LASSO_recipe <- recipe(Age ~ MaxHR + HeartDisease_bin, data = new_heart) |>
  step_dummy(HeartDisease_bin) |>
  step_normalize(MaxHR) |>
  step_interact( ~ MaxHR:starts_with("HeartDisease_bin_"))
LASSO_recipe
```

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



-- Inputs

Number of variables by role

outcome:   1
predictor: 2



-- Operations

* Dummy variables from: HeartDisease_bin

* Centering and scaling for: MaxHR

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

Selecting best model

```r
#Create folds
heart_folds <- vfold_cv(train, 10)

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

#Create Workflow
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
```

Fit the model

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

Pick the best

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

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)                    52.5  0.0000000001
2 MaxHR                          -4.26 0.0000000001
```

```
3 HeartDisease_bin_X1             2.76 0.0000000001
4 MaxHR_x_HeartDisease_bin_X1     2.05 0.0000000001
```

Without looking at the RMSE calculations, I would expect them to be about the same. This is because the penalty for the best LASSO model is very small for all the parameters, so the model isn't very different that the original model.

Comparing RMSE

```r
#RMSE for our best LASSO model
lasso_rmse <- LASSO_final |>
  predict(test) |>
  pull() |>
  rmse_vec(truth = test$Age)

#Compare to OLS
lasso_rmse
```

```
[1] 9.09553
```

```r
ols_rmse
```

```
[1] 9.100206
```

The RMSEs are roughly the same even though the parameters are different because they are both predicting the new data well, despite not being the exact same model type.

Task 6: Logistic Regression

```r
#Making the first model just the original basic one with interaction
LR1_model <- recipe(HeartDisease_bin ~ Age + MaxHR, data = new_heart) |>
  step_normalize(all_numeric())
LR2_model <- recipe(HeartDisease_bin ~ Age + RestingBP + Cholesterol + MaxHR,
                    data = new_heart) |>
  step_normalize(all_numeric())

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

```r
#Setting up workflows
LR1_wkf <- workflow() |>
  add_recipe(LR1_model) |>
  add_model(LR_spec)
LR2_wkf <- workflow() |>
  add_recipe(LR2_model) |>
  add_model(LR_spec)
```

Fit using CV folds and collect metrics

```r
LR1_fit <- LR1_wkf |>
  fit_resamples(heart_folds, metrics = metric_set(accuracy, mn_log_loss))
LR2_fit <- LR2_wkf |>
  fit_resamples(heart_folds, metrics = metric_set(accuracy, mn_log_loss))

rbind(LR1_fit |> collect_metrics(),
      LR2_fit |> collect_metrics()) |>
  mutate(Model = c("Model 1", "Model 1", "Model 2", "Model 2")) |>
  select(Model, everything())
```

```
# A tibble: 4 x 7
  Model   .metric     .estimator  mean     n std_err .config
  <chr>   <chr>       <chr>      <dbl> <int>   <dbl> <chr>
1 Model 1 accuracy    binary     0.684    10  0.0205 Preprocessor1_Model1
2 Model 1 mn_log_loss binary     0.595    10  0.0122 Preprocessor1_Model1
3 Model 2 accuracy    binary     0.693    10  0.0127 Preprocessor1_Model1
4 Model 2 mn_log_loss binary     0.584    10  0.0121 Preprocessor1_Model1
```

The second model, which has more predictors, is my model of choice because it has a higher accuracy and lower log loss than the other model.

Final fit

```r
LR_train_fit <- LR2_wkf |> fit(train)
predictions <- predict(LR_train_fit, test) |> pull()
confusionMatrix(data = predictions, reference = test$HeartDisease_bin)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
```

```
         0 70 21
         1 24 69

               Accuracy : 0.7554
                 95% CI : (0.6868, 0.8157)
    No Information Rate : 0.5109
    P-Value [Acc > NIR] : 8.729e-12

                  Kappa : 0.511

 Mcnemar's Test P-Value : 0.7656

            Sensitivity : 0.7447
            Specificity : 0.7667
         Pos Pred Value : 0.7692
         Neg Pred Value : 0.7419
             Prevalence : 0.5109
         Detection Rate : 0.3804
   Detection Prevalence : 0.4946
      Balanced Accuracy : 0.7557

       'Positive' Class : 0
```

The model does fairly well on the testing data, as we can see from the above confusion matrix. There are 21 false negatives and 24 false positives from the testing data. We also obtained sensitivity = 0.7447 and specificity = 0.7667 from the confusionMatrix() function above. The sensitivity value means that we were 74.47% accurate in predicting true positives (values that were predicted positive and were positive), and the specificity value means that we were 76.67% accurate in predicting true negatives (values that were predicted negative and were negative).