Homework 5: Heart Disease Model Comparison

Task 1: Conceptual Questions

- **Question 1: What is the purpose of using cross-validation when fitting a random forest model?**: The purpose of using cross-validation when fitting a random forest model is to estimate the model's generalization performance on unseen data. It helps us select optimal hyperparameters, such as the number of trees or the number of variables to try at each split (mtry), and avoid overfitting. Even though random forests reduce variance, cross-validation gives us a more stable estimate of how well the model will perform in practice.
- **Question 2: Describe the bagged tree algorithm.**: The bagged tree algorithm is an ensemble method that builds multiple decision trees using bootstrap samples of the training data. Each tree is trained on a different resampled dataset, and predictions are aggregated (averaged for regression, majority vote for classification). This process reduces the model's variance and improves stability, especially for high-variance models like decision trees.
- **Question 3: What is meant by a general linear model?**: A general linear model (GLM) refers to a statistical model that explains a continuous outcome variable as a linear combination of one or more predictor variables. It includes multiple linear regression, ANOVA, and ANCOVA models, assuming normally distributed errors and linear relationships.
- **Question 4: 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 in a multiple linear regression model allows the effect of one predictor variable on the outcome to depend on the value of another predictor. In other words, the relationship between a predictor and the response changes depending on the level of the interacting variable. Without the interaction term, the model assumes the effects of predictors are independent and additive.
- **Question 5: Why do we split our data into a training and test set?**: We split our data into training and test sets to evaluate how well our model generalizes to new,

unseen data. The training set is used to build and fit the model, while the test set is used to estimate the model's performance on future data. Without a separate test set, we risk overfitting and getting overly optimistic estimates of accuracy.

Task 2: Data Preparation

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

heart <- read_csv("heart.csv")

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
```

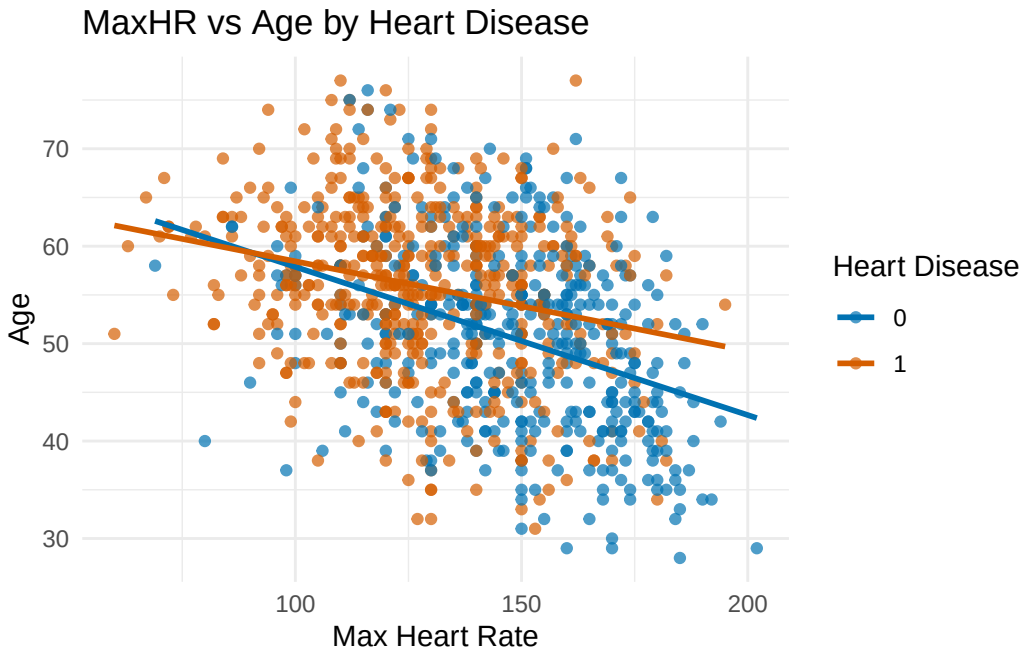
```
# Check variable type of HeartDisease
str(heart$HeartDisease)
```

```
num [1:918] 0 1 0 1 0 0 0 0 1 0 ...
```

```r
# Convert HeartDisease to a factor and remove ST_Slope
new_heart <- heart %>%
  mutate(HeartDisease_factor = as.factor(HeartDisease)) %>%
  select(-ST_Slope, -HeartDisease)
```

Task 3: Exploratory Data Analysis (EDA)

```r
ggplot(new_heart, aes(x = MaxHR, y = Age, color = HeartDisease_factor)) +
  geom_point(alpha = 0.7) +
  geom_smooth(method = "lm", se = FALSE) +
  scale_color_manual(values = c("0" = "#0072B2", "1" = "#D55E00")) +
  labs(title = "MaxHR vs Age by Heart Disease",
       x = "Max Heart Rate", y = "Age", color = "Heart Disease") +
  theme_minimal()
```



Based on the scatterplot, the relationship between Max Heart Rate (MaxHR) and Age differs depending on heart disease status. For individuals with heart disease, Age tends to remain

higher even at lower MaxHR, while those without heart disease show a stronger negative relationship between MaxHR and Age.

We used a **colorblind-friendly palette** (blue and orange) to distinguish between individuals with and without heart disease. Since the regression lines have visibly different slopes, an **interaction model** is more appropriate than an additive model. This allows us to model how the effect of MaxHR on Age depends on heart disease status.

Task 4: Train-Test Split

```
set.seed(101)
split <- initial_split(new_heart, prop = 0.8, strata = HeartDisease_factor)
train <- training(split)
test <- testing(split)
```

Task 5: OLS and LASSO

```
## 1. Fit an OLS interaction model (named ols_mlr)

ols_model <- linear_reg() %>%
  set_engine("lm")

ols_recipe <- recipe(Age ~ MaxHR + HeartDisease_factor, data = train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(terms = ~ MaxHR:starts_with("HeartDisease_factor"))

ols_mlr <- workflow() %>%
  add_model(ols_model) %>%
  add_recipe(ols_recipe)

ols_fit <- fit(ols_mlr, data = train)

# Extract and summarize the fitted model
ols_fit_lm <- extract_fit_parsnip(ols_fit)$fit
summary(ols_fit_lm)
```

```
Call:
stats::lm(formula = ..y ~ ., data = data)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-23.6887  -5.7711   0.5074   5.9777  22.3542

Coefficients:
                               Estimate Std. Error t value Pr(>|t|)
(Intercept)                    73.58712    3.09052  23.811  < 2e-16 ***
MaxHR                          -0.15396    0.02052  -7.503 1.82e-13 ***
HeartDisease_factor_X1         -7.30998    3.92591  -1.862   0.0630 .
MaxHR_x_HeartDisease_factor_X1  0.07058    0.02774   2.545   0.0111 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.645 on 730 degrees of freedom
Multiple R-squared:  0.1569,    Adjusted R-squared:  0.1535
F-statistic:  45.3 on 3 and 730 DF,  p-value: < 2.2e-16
```

```r
## 2. Evaluate predictive performance on the test set using RMSE

ols_preds <- predict(ols_fit, new_data = test) %>%
  bind_cols(test)

rmse(ols_preds, truth = Age, estimate = .pred)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard        8.45
```

```r
## 3. Fit a LASSO interaction model using cross-validation

# Define LASSO recipe: standardize predictors, dummy-code factors, and include interaction
lasso_recipe <- recipe(Age ~ MaxHR + HeartDisease_factor, data = train) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(terms = ~ MaxHR:starts_with("HeartDisease_factor"))


# Print the LASSO recipe to show the preprocessing steps
tidy(lasso_recipe)
```

```
# A tibble: 3 x 6
  number operation type      trained skip  id
   <int> <chr>     <chr>     <lgl>   <lgl> <chr>
1      1 step      normalize FALSE   FALSE normalize_3GOVe
2      2 step      dummy     FALSE   FALSE dummy_oPwFp
3      3 step      interact  FALSE   FALSE interact_4o8i1
```

```r
# Also show the variables created after prepping the recipe
prep(lasso_recipe) %>%
  summary()
```

```
# A tibble: 4 x 4
  variable                       type      role      source
  <chr>                          <list>    <chr>     <chr>
1 MaxHR                          <chr [2]> predictor original
2 Age                            <chr [2]> outcome   original
3 HeartDisease_factor_X1         <chr [2]> predictor derived
4 MaxHR_x_HeartDisease_factor_X1 <chr [2]> predictor derived
```

```r
# Specify the LASSO model (mixture = 1 for LASSO)
lasso_model <- linear_reg(penalty = tune(), mixture = 1) %>%
  set_engine("glmnet")

# Create workflow
lasso_wf <- workflow() %>%
  add_model(lasso_model) %>%
  add_recipe(lasso_recipe)

# Perform 10-fold cross-validation to tune lambda
set.seed(123)
folds <- vfold_cv(train, v = 10)
grid <- grid_regular(penalty(), levels = 200)

tune_res <- tune_grid(
  lasso_wf,
  resamples = folds,
  grid = grid,
  metrics = metric_set(rmse)
)

# Select the best lambda and finalize the workflow
```

```r
best_lasso <- select_best(tune_res, metric = "rmse")
final_lasso <- finalize_workflow(lasso_wf, best_lasso)

# Fit the final model
lasso_fit <- fit(final_lasso, data = train)

## 4. Evaluate LASSO model on the test set

lasso_preds <- predict(lasso_fit, new_data = test) %>%
  bind_cols(test)

rmse(lasso_preds, truth = Age, estimate = .pred)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard        8.45
```

```r
## 5. Inspect selected variables using tidy()
lasso_fit_model <- extract_fit_parsnip(lasso_fit)$fit

# View only the final step with non-zero coefficients
tidy(lasso_fit_model) %>%
  filter(step == max(step)) %>%
  filter(estimate != 0)
```

```
# A tibble: 4 x 5
  term                           step estimate  lambda dev.ratio
  <chr>                         <dbl>    <dbl>   <dbl>     <dbl>
1 (Intercept)                      66    52.5  0.00815     0.157
2 MaxHR                            66    -3.88 0.00815     0.157
3 HeartDisease_factor_X1           66     2.36 0.00815     0.157
4 MaxHR_x_HeartDisease_factor_X1   66     1.74 0.00815     0.157
```

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 RMSEs to be quite similar because the number of predictors is small and the variables are all relevant to predicting Age. LASSO may slightly shrink coefficients, but it's unlikely to exclude variables entirely in this setting.

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

The RMSE for both models was approximately 8.45, indicating that the predictive performance of the two models is nearly identical.

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

Although LASSO introduces shrinkage and potentially variable selection, in this case it retained the same set of informative predictors as the OLS model. As a result, both models yield similar predictions on the test data.

Task 6: Logistic Regression

```
# Model 1: Age + MaxHR + Cholesterol + Sex
log_recipe1 <- recipe(HeartDisease_factor ~ Age + MaxHR + Cholesterol + Sex, data = train)

log_model <- logistic_reg() %>%
  set_engine("glm")

log_wf1 <- workflow() %>%
  add_model(log_model) %>%
  add_recipe(log_recipe1)

# Set up 10-fold cross-validation with 3 repeats
set.seed(123)
cv <- vfold_cv(train, v = 10, repeats = 3)

# Fit Model 1 with cross-validation
log_cv1 <- fit_resamples(
  log_wf1,
  resamples = cv,
  metrics = metric_set(roc_auc, accuracy)
)

# Model 2: Age + Cholesterol + Sex + ChestPainType
log_recipe2 <- recipe(HeartDisease_factor ~ Age + Cholesterol + Sex + ChestPainType, data

log_wf2 <- workflow() %>%
  add_model(log_model) %>%
  add_recipe(log_recipe2)
```

```r
# Fit Model 2 with cross-validation
log_cv2 <- fit_resamples(
  log_wf2,
  resamples = cv,
  metrics = metric_set(roc_auc, accuracy)
)

# Compare performance of the two models
collect_metrics(log_cv1)
```

```
# A tibble: 2 x 6
  .metric  .estimator  mean     n std_err .config
  <chr>    <chr>      <dbl> <int>   <dbl> <chr>
1 accuracy binary     0.725    30 0.00874 Preprocessor1_Model1
2 roc_auc  binary     0.789    30 0.00982 Preprocessor1_Model1
```

```r
collect_metrics(log_cv2)
```

```
# A tibble: 2 x 6
  .metric  .estimator  mean     n std_err .config
  <chr>    <chr>      <dbl> <int>   <dbl> <chr>
1 accuracy binary     0.789    30 0.00905 Preprocessor1_Model1
2 roc_auc  binary     0.846    30 0.00824 Preprocessor1_Model1
```

Model Selection Summary

Model 2 (using ChestPainType instead of MaxHR) achieved higher cross-validation performance: - Accuracy: 0.789 - ROC AUC: 0.846

Therefore, we chose Model 2 as the better performing model.

Final Evaluation on Test Set

```r
# Fit the selected model (Model 2) on full training data
log_final_fit <- fit(log_wf2, data = train)

# Predict on test set
log_preds <- predict(log_final_fit, new_data = test, type = "class") %>%
```

```
  bind_cols(test)

# Evaluate with confusion matrix
conf_mat <- confusionMatrix(log_preds$.pred_class, test$HeartDisease_factor)
conf_mat
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 56 16
         1 26 86

               Accuracy : 0.7717
                 95% CI : (0.7042, 0.8303)
    No Information Rate : 0.5543
    P-Value [Acc > NIR] : 6.886e-10

                  Kappa : 0.5324

 Mcnemar's Test P-Value : 0.1649

            Sensitivity : 0.6829
            Specificity : 0.8431
         Pos Pred Value : 0.7778
         Neg Pred Value : 0.7679
             Prevalence : 0.4457
         Detection Rate : 0.3043
   Detection Prevalence : 0.3913
      Balanced Accuracy : 0.7630

       'Positive' Class : 0
```

Interpretation of Results

- **Accuracy**: 77.2%
- **Sensitivity**: 68.3%
- **Specificity**: 84.3%

Interpretation: - **Sensitivity (68.3%)** means 68.3% of patients with heart disease were correctly identified. - **Specificity (84.3%)** means 84.3% of patients without heart disease were correctly identified.

This suggests that the model performs well at distinguishing patients with and without heart disease, with a stronger ability to correctly identify those without the disease.