Predictive Modeling

Calista Harris

Task 1: Conceptual Questions

Question 1

Cross-validation helps us estimate how well a random forest model is likely to perform on unseen data by repeatedly splitting the data into training and validation sets. This process reduces variability in the performance estimate that could arise from a single train/test split and allows for more reliable model assessment. Although random forests have built-in error estimation through out-of-bag observations, cross-validation can provide a complementary or alternative method, especially when comparing multiple models or tuning hyperparameters.

Question 2

The bagged tree algorithm, short for bootstrap aggregation, involves generating multiple bootstrap samples from the original dataset, fitting a decision tree to each sample, and aggregating their predictions. For regression tasks, the final prediction is the average of individual tree predictions, while for classification, it is typically determined by majority vote. This ensemble method helps reduce variance and improves predictive performance compared to a single tree model.

Question 3

A general linear model (GLM) is a statistical model where the expected value of the response variable is modeled as a linear combination of the explanatory variables. It has the form $E(Y|X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$, and is used for tasks such as multiple linear regression and ANOVA. The model assumes the errors are normally distributed and have constant variance, making it a foundational approach for modeling relationships between variables.

Question 4

Adding an interaction term to a multiple linear regression model allows the effect of one explanatory variable on the response variable to depend on the level of another explanatory variable. This enables the model to capture relationships where the combined effect of two variables is not merely additive. In mathematical terms, it adds a term like $\beta_3 X_1 X_2$ to the model, allowing the slope of one variable to change based on the value of the other.

Question 5

Splitting the data into a training and test set allows us to evaluate how well a model generalizes to new, unseen data. The training set is used to fit the model, while the test set provides an unbiased estimate of the model's predictive performance. This helps guard against overfitting and ensures that the model is not simply memorizing the training data, but instead learning patterns that can apply more broadly.

Task 2: Data Prep

Packages and Data

```
#load require libraries
library(tidyverse)
library(tidymodels)
library(caret)
library(yardstick)

#read in the heart disease dataset as a tibble
heart <- read_csv("data/heart.csv") |>
  as_tibble()
```

Question 1

```
#summarize the data
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. According to the summary() output, `HeartDisease` is currently treated as a numeric variable in R. This is evident from the statistical summaries displayed — Min, 1st Qu., Median, Mean, etc. — all indicators of a quantitative numeric type.

b. No, this does not make sense for modeling. The `HeartDisease` variable encodes binary outcomes — either 0 (no heart disease) or 1 (presence of heart disease). As described in Logistic Regression Models, binary outcomes should be treated as categorical when modeling classification problems. Using it as numeric may lead to inappropriate modeling choices, such as applying linear regression when logistic regression is more appropriate.

Question 2

```r
#convert HeartDisease to a factor (categorical) variable and rename it
new_heart <- heart |>
  mutate(HeartDisease_status = factor(HeartDisease)) |>
  #drop the original numeric HeartDisease variable and the ST_Slope variable
  select(-ST_Slope, -HeartDisease)

#view the structure of the updated data set
glimpse(new_heart)
```

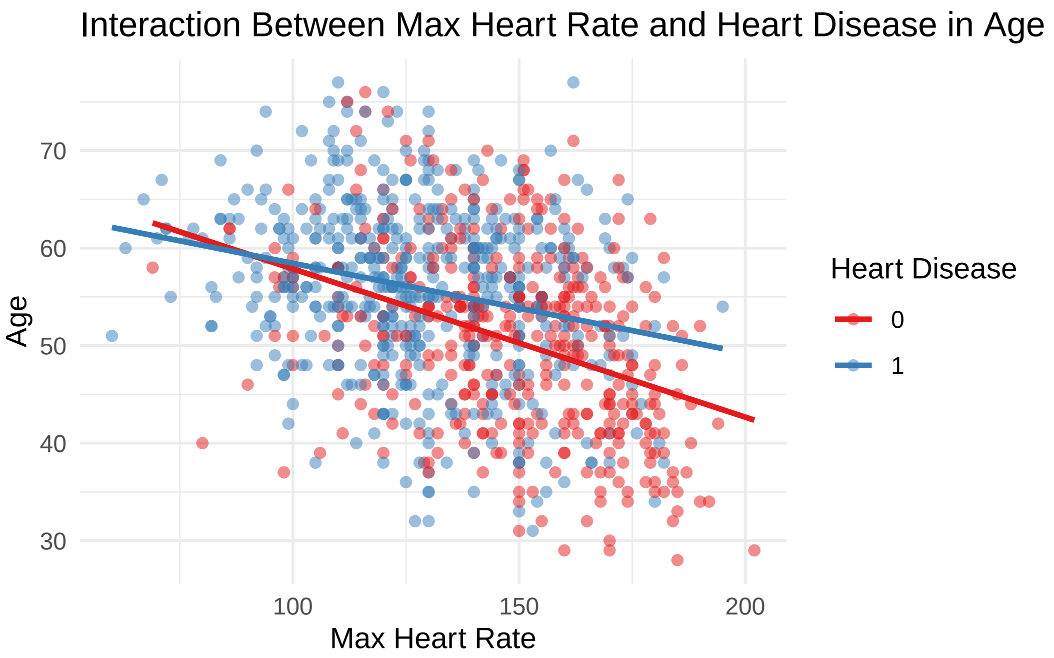
```
Rows: 918
Columns: 11
$ Age                 <dbl> 40, 49, 37, 48, 54, 39, 45, 54, 37, 48, 37, 58, 39~
$ Sex                 <chr> "M", "F", "M", "F", "M", "M", "F", "M", "M", "F", ~
$ ChestPainType       <chr> "ATA", "NAP", "ATA", "ASY", "NAP", "NAP", "ATA", "~
$ RestingBP           <dbl> 140, 160, 130, 138, 150, 120, 130, 110, 140, 120, ~
$ Cholesterol         <dbl> 289, 180, 283, 214, 195, 339, 237, 208, 207, 284, ~
$ FastingBS           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ RestingECG          <chr> "Normal", "Normal", "ST", "Normal", "Normal", "Nor~
$ MaxHR               <dbl> 172, 156, 98, 108, 122, 170, 170, 142, 130, 120, 1~
$ ExerciseAngina      <chr> "N", "N", "N", "Y", "N", "N", "N", "N", "Y", "N", ~
$ Oldpeak             <dbl> 0.0, 1.0, 0.0, 1.5, 0.0, 0.0, 0.0, 0.0, 1.5, 0.0, ~
$ HeartDisease_status <fct> 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1,~
```

Task 3: EDA

Question 1

```r
#create the scatterplot with separate trend lines by heart disease
ggplot(new_heart, aes(x = MaxHR, y = Age, color = HeartDisease_status)) +
  geom_point(alpha = 0.5) +
  #remove the standard error bars
  geom_smooth(method = "lm", se = FALSE) +
  #Set1 is color-blind friendly
  scale_color_brewer(palette = "Set1", name = "Heart Disease") +
  labs(
    title = "Interaction Between Max Heart Rate and Heart Disease in Age",
    x = "Max Heart Rate",
    y = "Age"
  ) +
  theme_minimal()
```

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



Question 2

The scatterplot shows non-parallel trend lines for people with and without heart disease, indicating that the relationship between Max Heart Rate and Age differs by Heart Disease. Specifically, the slope for individuals without heart disease is steeper than for those with heart disease. This suggests an interaction effect, where the impact of `MaxHR` on `Age` depends on `HeartDisease_status`. Recall Multiple Linear Regression, interaction terms allow the slope of one variable to change based on the level of another. Therefore, an interaction model is more appropriate than an additive model for this analysis.

Task 4: Testing and Training

```r
#set seed for reproducibility
set.seed(101)

#perform 80-20 train-test split
heart_split <- initial_split(new_heart, prop = 0.8)

#create training and test data sets
train <- training(heart_split)
test <- testing(heart_split)

#check sizes of each set
nrow(train)
```

```
[1] 734
```

```r
nrow(test)
```

```
[1] 184
```

Task 5: OLS and LASSO

Question 1

```r
#fit OLS model with interaction between MaxHR and HeartDisease
ols_mlr <- lm(Age ~ MaxHR * HeartDisease_status, data = train)

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

```
Call:
lm(formula = Age ~ MaxHR * HeartDisease_status, 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_status1       -8.58502    3.83433  -2.239  0.02546 *  
MaxHR:HeartDisease_status1  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
#predict response (Age) on test data using the OLS model
pred_ols <- predict(ols_mlr, newdata = test)

#bind predictions to test data for RMSE evaluation
ols_results <- test |>
  mutate(pred = pred_ols)
```

```
#compute RMSE using yardstick
ols_rmse <- rmse(ols_results, truth = Age, estimate = pred)
ols_rmse
```

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

Question 3

```
#(a-d) Define the recipe with normalization, dummy coding, and interaction
LASSO_recipe <- recipe(Age ~ MaxHR + HeartDisease_status, data = train) |>
  step_normalize(all_numeric_predictors()) |>   #(b) standardize
  step_dummy(all_nominal_predictors()) |>       #dummy encode
  step_interact(terms = ~ MaxHR:starts_with("HeartDisease_status")) #(c & d)

#print recipe
LASSO_recipe
```

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




-- Inputs


Number of variables by role


outcome:   1
predictor: 2




-- Operations
```

```
* Centering and scaling for: all_numeric_predictors()

* Dummy variables from: all_nominal_predictors()

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

Question 4

```r
#specify LASSO model
LASSO_spec <- linear_reg(penalty = tune(), mixture = 1) |>
  set_engine("glmnet")

#combine recipe and model
LASSO_wflow <- workflow() |>
  add_recipe(LASSO_recipe) |>
  add_model(LASSO_spec)

#cross-validation folds
set.seed(101)
LASSO_folds <- vfold_cv(train, v = 10)

#tune penalty on 10-fold CV
LASSO_tuned <- tune_grid(
  object = LASSO_wflow,
  resamples = LASSO_folds,
  grid = 25,                    #try 25 penalty values
  metrics = metric_set(rmse)
)
```

```
Warning: package 'glmnet' was built under R version 4.5.1
```

```r
#select best-performing penalty
best_penalty <- select_best(LASSO_tuned, metric = "rmse")

#finalize workflow
final_LASSO <- finalize_workflow(LASSO_wflow, best_penalty)

#fit to training data
LASSO_fit <- fit(final_LASSO, data = train)
```

```
#report the results
tidy(LASSO_fit)
```

```
# A tibble: 4 x 3
  term                          estimate  penalty
  <chr>                            <dbl>    <dbl>
1 (Intercept)                      52.5  1.57e-10
2 MaxHR                            -4.26 1.57e-10
3 HeartDisease_status_X1            2.76 1.57e-10
4 MaxHR_x_HeartDisease_status_X1    2.05 1.57e-10
```

Question 5

Based on the output from the LASSO model in Question 4, we would expect the RMSE to be roughly similar to that of the OLS model. This is because all variables—including the interaction between `MaxHR` and `HeartDisease_status` have non-zero coefficients in the LASSO model. Since LASSO didn't shrink any coefficients to zero (which it often does for variable selection), it suggests that the model complexity and fit are comparable to OLS. Therefore, the predictive performance, as measured by RMSE, is likely to be similar.

Question 6

```
#generate predictions on the test set using the final LASSO model
LASSO_preds <- predict(LASSO_fit, new_data = test) |>
  bind_cols(test)  #attach predictions to the actual test data

#calculate RMSE for the LASSO model using yardstick
LASSO_rmse <- rmse(LASSO_preds, truth = Age, estimate = .pred)

#create a tibble comparing RMSE between OLS and LASSO models
compare_rmse <- tibble(
  Model = c("OLS", "LASSO"),
  RMSE = c(ols_rmse$.estimate, LASSO_rmse$.estimate)
)

#display the RMSE comparison
compare_rmse
```

```
# A tibble: 2 x 2
  Model  RMSE
  <chr> <dbl>
1 OLS    9.10
2 LASSO  9.10
```

Question 7

Even though the OLS and LASSO models have different coefficient estimates, their RMSE values are roughly the same. This is because both models capture the same underlying relationship between `Age`, `MaxHR`, and `HeartDisease_status`. LASSO applies a penalty that shrinks coefficients, while OLS does not. In our results, the key predictors and their interaction were retained in both models. The LASSO coefficients are smaller due to regularization, but the predictive patterns are similar. LASSO aims to improve model generalization without drastically changing predictions when the signal is strong. Therefore, both models yield similar RMSE on the test set despite differing coefficients.

Task 6: Logistic Regression

Question 1

```r
#propose two logistic regression models

#set seed for reproducibility
set.seed(101)

#define cross-validation: repeated 10-fold CV (3 repeats)
ctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

#model 1: Uses Age and MaxHR
model_1 <- train(
  HeartDisease_status ~ Age + MaxHR,
  data = train,
  method = "glm",
  family = "binomial",
  trControl = ctrl
)

#model 2: Uses Age, MaxHR, and ExerciseAngina
model_2 <- train(
  HeartDisease_status ~ Age + MaxHR + ExerciseAngina,
  data = train,
  method = "glm",
  family = "binomial",
  trControl = ctrl
)

#identify the best performing model

# Compare model performance
model_1$resample
```

```
    Accuracy     Kappa    Resample
1  0.6164384 0.2102009 Fold01.Rep1
2  0.7534247 0.4777424 Fold02.Rep1
3  0.6891892 0.3644511 Fold03.Rep1
4  0.7123288 0.4087929 Fold04.Rep1
5  0.6438356 0.2521671 Fold05.Rep1
```

```
6  0.7567568 0.4969789 Fold06.Rep1
7  0.7027027 0.3805175 Fold07.Rep1
8  0.6250000 0.2199037 Fold08.Rep1
9  0.7297297 0.4368341 Fold09.Rep1
10 0.6351351 0.2483070 Fold10.Rep1
11 0.7260274 0.4393241 Fold01.Rep2
12 0.6027397 0.1971938 Fold02.Rep2
13 0.7123288 0.4014057 Fold03.Rep2
14 0.7123288 0.4087929 Fold04.Rep2
15 0.7027027 0.3805175 Fold05.Rep2
16 0.6756757 0.3190184 Fold06.Rep2
17 0.6712329 0.3214562 Fold07.Rep2
18 0.7432432 0.4710309 Fold08.Rep2
19 0.6575342 0.2772277 Fold09.Rep2
20 0.6621622 0.2823894 Fold10.Rep2
21 0.6849315 0.3524875 Fold01.Rep3
22 0.6438356 0.2711214 Fold02.Rep3
23 0.6575342 0.2714571 Fold03.Rep3
24 0.6756757 0.3137558 Fold04.Rep3
25 0.6438356 0.2666151 Fold05.Rep3
26 0.7027027 0.3898051 Fold06.Rep3
27 0.6486486 0.2622699 Fold07.Rep3
28 0.6621622 0.2879138 Fold08.Rep3
29 0.8219178 0.6308829 Fold09.Rep3
30 0.7397260 0.4732245 Fold10.Rep3
```

```r
model_2$resample
```

```
    Accuracy     Kappa    Resample
1  0.7702703 0.5337287 Fold01.Rep1
2  0.6712329 0.3544584 Fold02.Rep1
3  0.7837838 0.5627770 Fold03.Rep1
4  0.7567568 0.5152838 Fold04.Rep1
5  0.6805556 0.3510972 Fold05.Rep1
6  0.7534247 0.4957790 Fold06.Rep1
7  0.7837838 0.5595238 Fold07.Rep1
8  0.7808219 0.5589124 Fold08.Rep1
9  0.7123288 0.4137667 Fold09.Rep1
10 0.8648649 0.7267356 Fold10.Rep1
11 0.7534247 0.4996192 Fold01.Rep2
12 0.7567568 0.5081241 Fold02.Rep2
13 0.8378378 0.6720827 Fold03.Rep2
```

```
14 0.7567568 0.5081241 Fold04.Rep2
15 0.7945205 0.5812620 Fold05.Rep2
16 0.7083333 0.4255319 Fold06.Rep2
17 0.8630137 0.7172734 Fold07.Rep2
18 0.6986301 0.4043027 Fold08.Rep2
19 0.7432432 0.4827079 Fold09.Rep2
20 0.6756757 0.3392857 Fold10.Rep2
21 0.7702703 0.5405405 Fold01.Rep3
22 0.7123288 0.4332717 Fold02.Rep3
23 0.6891892 0.3691623 Fold03.Rep3
24 0.7702703 0.5302465 Fold04.Rep3
25 0.8356164 0.6664128 Fold05.Rep3
26 0.7162162 0.4282561 Fold06.Rep3
27 0.7808219 0.5552171 Fold07.Rep3
28 0.8055556 0.6096050 Fold08.Rep3
29 0.7837838 0.5562219 Fold09.Rep3
30 0.7123288 0.4087929 Fold10.Rep3
```

```r
# Get average accuracy
mean(model_1$resample$Accuracy)
```

```
[1] 0.6870495
```

```r
mean(model_2$resample$Accuracy)
```

```
[1] 0.7574132
```

Based on repeated cross-validation accuracy, Model 2 performs slightly better than Model 1. This suggests that adding `ExerciseAngina` improves the model's ability to classify heart disease. Recall logistic regression, including categorical predictors that are meaningfully associated with the outcome can increase model performance.

Question 2

```r
#predict on test set using the better model (Model 2)
pred <- predict(model_2, newdata = test)

#confusion matrix
confusionMatrix(pred, test$HeartDisease_status)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 72 29
         1 22 61

               Accuracy : 0.7228
                 95% CI : (0.6522, 0.7861)
    No Information Rate : 0.5109
    P-Value [Acc > NIR] : 3.683e-09

                  Kappa : 0.4445

 Mcnemar's Test P-Value : 0.4008

            Sensitivity : 0.7660
            Specificity : 0.6778
         Pos Pred Value : 0.7129
         Neg Pred Value : 0.7349
             Prevalence : 0.5109
         Detection Rate : 0.3913
   Detection Prevalence : 0.5489
      Balanced Accuracy : 0.7219

       'Positive' Class : 0
```

Question 3

From the confusion matrix, the sensitivity is 0.7660, and the specificity is 0.6778. Sensitivity measures how well the model correctly identifies individuals without heart disease (the positive class was set as 0). A sensitivity of 76.6% means that about three-quarters of healthy individuals were correctly predicted. Specificity, on the other hand, measures the model's ability to correctly detect individuals with heart disease, 67.8% of actual heart disease cases were accurately classified. Recall Logistic Regression Models, sensitivity and specificity are important diagnostic metrics, especially in healthcare. Sensitivity helps minimize false negatives, which is critical when failing to detect disease has serious consequences, while specificity reduces false positives, avoiding unnecessary stress or treatment.