Homework5

Jessie Heise

Task 1: Conceptual Questions

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

To avoid overfitting.

2. Describe the bagged tree algorithm.

The bagged tree algorithm fits many trees on bootstrap samples and combines the predictions.

3. What is meant by a general linear model?

A statistical model used on non-normal data.

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?

Allows for the effect of one variable to depend on the value of another.

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

We want to be able to use models to predict well for observations it has yet to see and avoid overfitting the model based on the test data which is why the training set is used to traing the model and the test set is used to judge effectiveness of the model.

Task 2: Data Prep

Packages & Library

Run a report summary()

```
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  
```

Report summary questions

1. What type of variable (in R) is Heart Disease? Categorical or Quantitative?

Numeric. Categorical.

2. Does this make sense? Why or why not.

This does not make sense because the heart disease variable tells whether or not someone has heart disease- essentially a yes or a no. This data should not be numeric.

Continued data prep

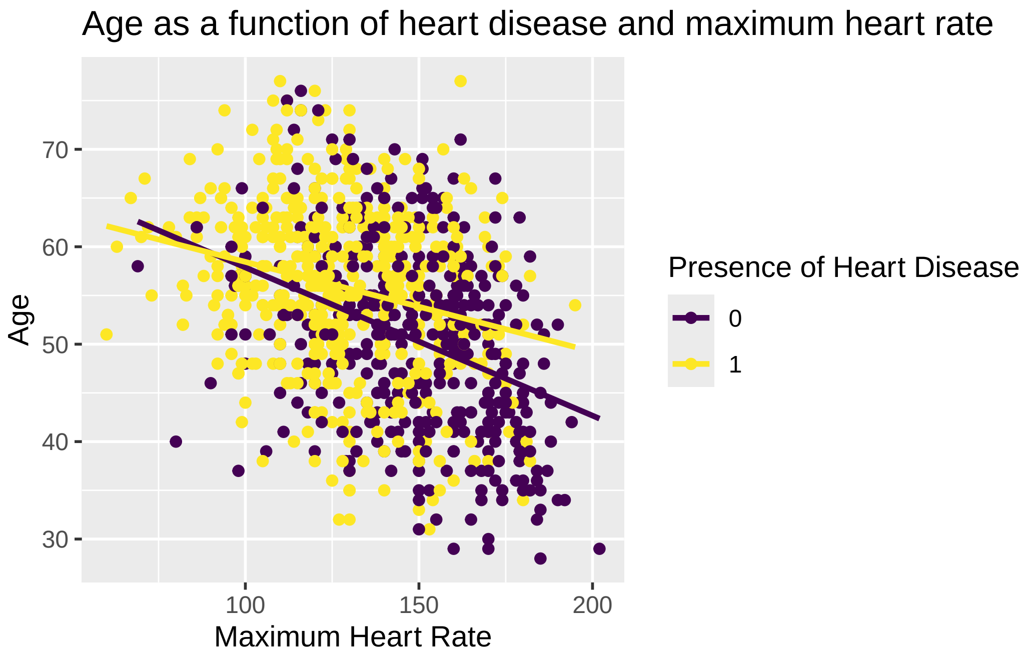
```r
# Change HeartDisease to logical, drop ST_Slope, HeartDisease
new_heart <- heart_data |>
  mutate(HeartDisease_pres = as.factor(HeartDisease)) |>
  select(-HeartDisease, -ST_Slope)
```

Task 3: EDA

Model someone's age as a function of heart disease and their max heart rate

```r
# Create appropriate scatterplot to visualize this relationship
ggplot2::ggplot(data = new_heart, aes(x= MaxHR, y = Age,
                                      color = HeartDisease_pres)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(
    title = "Age as a function of heart disease and maximum heart rate",
    x = "Maximum Heart Rate",
    y = "Age",
    color = "Presence of Heart Disease"
  ) +
  scale_color_viridis_d()
```

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



Based on visual evidence, do you think an interaction model or an additive model is more appropriate?

Based on the visual evidence, an interactive model is more appropriate because the lines for presence of heart disease cross, suggesting an interactive effect.

Task 4: Testing & Training

```
# Split data into a training and test set. Seed = 101. 80-20 split.
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

Fit an interaction model

```r
# Fit an interaction model with age as response,
# max hr + heart disease as explanatory variables
# using the training data set using OLS regression.
# report summary output
ols_mlr <- lm(Age ~ HeartDisease_pres * MaxHR, data = train)
summary(ols_mlr)
```

```
Call:
lm(formula = Age ~ HeartDisease_pres * MaxHR, 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 ***
HeartDisease_pres1       -8.58502    3.83433  -2.239  0.02546 *  
MaxHR                    -0.16992    0.02064  -8.233 8.43e-16 ***
HeartDisease_pres1:MaxHR  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
# Test model on testing data set
ols_predict <- predict(ols_mlr, newdata = test)

# Calculate RMSE for OLS
rmse_vec(test$Age, predict(ols_mlr, newdata = test))
```

```
[1] 9.100206
```

See if LASSO has better predictive performance

```r
# Use CV to select best tuning parameter.
# Evaluate LASSO model on testing data set
heart_CV_folds <- vfold_cv(new_heart, 10)

my_metrics <- metric_set(rmse)

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

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

-- Inputs

Number of variables by role

outcome:   1
predictor: 2

-- Operations

* Dummy variables from: HeartDisease_pres

* Centering and scaling for: MaxHR

* Interactions with: starts_with("HeartDisease_pres"):starts_with("MaxHR_")
```

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

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
```

```r
#warning will occur for one value of the tuning parameter, safe to ignore
LASSO_grid <- LASSO_wkf |>
  tune_grid(resamples = heart_CV_folds,
            grid = grid_regular(penalty(), levels = 5),
            metrics = my_metrics)
```

```
Warning: package 'glmnet' was built under R version 4.4.3
```

```r
LASSO_grid
```

```
# Tuning results
# 10-fold cross-validation
```

```
# A tibble: 10 x 4
   splits           id     .metrics         .notes
   <list>           <chr>  <list>           <list>
 1 <split [826/92]> Fold01 <tibble [5 x 5]> <tibble [0 x 3]>
 2 <split [826/92]> Fold02 <tibble [5 x 5]> <tibble [0 x 3]>
 3 <split [826/92]> Fold03 <tibble [5 x 5]> <tibble [0 x 3]>
 4 <split [826/92]> Fold04 <tibble [5 x 5]> <tibble [0 x 3]>
 5 <split [826/92]> Fold05 <tibble [5 x 5]> <tibble [0 x 3]>
 6 <split [826/92]> Fold06 <tibble [5 x 5]> <tibble [0 x 3]>
 7 <split [826/92]> Fold07 <tibble [5 x 5]> <tibble [0 x 3]>
 8 <split [826/92]> Fold08 <tibble [5 x 5]> <tibble [0 x 3]>
 9 <split [827/91]> Fold09 <tibble [5 x 5]> <tibble [0 x 3]>
10 <split [827/91]> Fold10 <tibble [5 x 5]> <tibble [0 x 3]>
```

```r
LASSO_grid[1, ".metrics"][[1]]
```

```
[[1]]
# A tibble: 5 x 5
       penalty .metric .estimator .estimate .config
         <dbl> <chr>   <chr>          <dbl> <chr>
1 0.0000000001 rmse    standard        9.87 Preprocessor1_Model1
2 0.0000000316 rmse    standard        9.87 Preprocessor1_Model2
3 0.00001      rmse    standard        9.87 Preprocessor1_Model3
4 0.00316      rmse    standard        9.87 Preprocessor1_Model4
5 1            rmse    standard        9.96 Preprocessor1_Model5
```

```r
LASSO_grid |>
  collect_metrics() |>
  filter(.metric == "rmse")
```

```
# A tibble: 5 x 7
       penalty .metric .estimator  mean     n std_err .config
         <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1 0.0000000001 rmse    standard    8.62    10   0.188 Preprocessor1_Model1
2 0.0000000316 rmse    standard    8.62    10   0.188 Preprocessor1_Model2
3 0.00001      rmse    standard    8.62    10   0.188 Preprocessor1_Model3
4 0.00316      rmse    standard    8.62    10   0.188 Preprocessor1_Model4
5 1            rmse    standard    8.70    10   0.177 Preprocessor1_Model5
```

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

lrmse <- as.numeric(lowest_rmse$penalty)


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

```
# A tibble: 3 x 3
  term                 estimate      penalty
  <chr>                   <dbl>        <dbl>
1 (Intercept)             51.9  0.0000000001
2 MaxHR                   -3.01 0.0000000001
3 HeartDisease_pres_X1     2.88 0.0000000001
```

I would expect the RMSE calculations to be different since the intercepts of the models are different.

Compare RMSEs:

```
ols_mlr |>
  predict(test) |>
  rmse_vec(truth = test$Age)
```

```
[1] 9.100206
```

```
LASSO_final |>
  predict(test) |>
  pull() |>
  rmse_vec(truth = test$Age)
```

```
[1] 8.999952
```

I think the RMSE calculations are roughly the same even though the coefficients for each model are different because both models might average the same errors.

Task 6: Logistic Regression

```
set.seed(3557)
heart_data <- heart_data |>
  mutate(HeartDisease = factor(HeartDisease))
heart_split <- initial_split(heart_data, prop = 0.8)
heart_train <- training(heart_split)
heart_test <- testing(heart_split)
heart_CV_folds <- vfold_cv(heart_train, 10)

LR1_rec <- recipe(HeartDisease ~ RestingBP + RestingECG,
                  data = heart_train) |>
  step_normalize(all_numeric(), -HeartDisease) |>
  step_dummy(RestingECG)
LR2_rec <- recipe(HeartDisease ~ ChestPainType + MaxHR + ExerciseAngina,
                  data = heart_train) |>
  step_normalize(all_numeric(), -HeartDisease) |>
  step_dummy(ChestPainType, ExerciseAngina)
LR2_rec |>
  prep(heart_train) |>
  bake(heart_train) |>
  colnames()
```

```
[1] "MaxHR"             "HeartDisease"      "ChestPainType_ATA"
[4] "ChestPainType_NAP" "ChestPainType_TA"  "ExerciseAngina_Y"
```

```
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(),
```

```r
     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     0.541    10 0.0156  Preprocessor1_Model1
2 Model1 mn_log_loss binary     0.681    10 0.00647 Preprocessor1_Model1
3 Model2 accuracy    binary     0.787    10 0.0165  Preprocessor1_Model1
4 Model2 mn_log_loss binary     0.453    10 0.0164  Preprocessor1_Model1
```

```r
mean(heart_train$HeartDisease == "1")
```

```
[1] 0.5599455
```

The best performing model is Model 2 because it has higher accuracy and log loss closer to 0. Model 2 uses Chest Pain Type, Max HR, and Exercise in Angina.

```r
final_model <- LR2_wkf |>
  fit(heart_data)
tidy(final_model)
```

```
# A tibble: 6 x 5
  term              estimate std.error statistic  p.value
  <chr>                <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)          0.481    0.139       3.47 5.22e- 4
2 MaxHR               -0.508    0.0941     -5.40 6.64e- 8
3 ChestPainType_ATA   -2.38     0.264      -9.03 1.68e-19
4 ChestPainType_NAP   -1.44     0.204      -7.04 1.90e-12
5 ChestPainType_TA    -0.735    0.340      -2.16 3.09e- 2
6 ExerciseAngina_Y     1.59     0.193       8.21 2.15e-16
```

```r
LR_train_fit <- LR2_wkf |>
  fit(heart_train)
LR2_wkf |>
  last_fit(heart_split, metrics = metric_set(accuracy, mn_log_loss)) |>
  collect_metrics()
```

```
# A tibble: 2 x 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary         0.745 Preprocessor1_Model1
2 mn_log_loss binary         0.514 Preprocessor1_Model1
```

Check how well Model 2 does on the test set using the confusionMatrix() function:

```r
LR2_test_fit <- LR2_wkf |>
  fit(heart_test)

conf_mat(heart_test |>
           mutate(estimate = LR2_test_fit |>
                    predict(heart_test) |>
                    pull()), HeartDisease, estimate)
```

```
          Truth
Prediction  0  1
         0 58 16
         1 29 81
```

```r
TP <- 81
FN <- 16
TN <- 58
FP <- 29

sensitivity <- TP/(TP+FN)
specificity <- TN/(TN+FP)

print(sensitivity)
```

```
[1] 0.8350515
```

```r
print(specificity)
```

```
[1] 0.6666667
```

Since sensitivity is the true positive rate, the model accurately predicts the presence of heart disease in 83.5% of the time. Since Specificity is the true negative rate, the model accurately predicts the absence of heart disease 66.7% of hte time. The model is better at predicting heart disease than not predicting it.