Homework5

Holly Probasco

Task 1: Conceptual Questions

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

- The purpose is to help choose the best model within the random forest selection process. It shows the error for the models used and also the accuracy of the predictions.

2. Describe the bagged tree algorithm.

- The process involves creating many bootstrap resamples. Each resample creates a new tree. Then, the predictions from each tree are combined in some way to then create the final prediction. For regression trees this is taking the mean, and for classification trees the most common prediction is used.

3. What is meant by a general linear model?

- A general linear model is a linear model that is able to be used on multiple predictors, which includes categorical variables and also interaction terms.

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?

- Interaction terms allow one predictor to be dependant on another, allowing for more model flexibility. Otherwise, each of the terms in the model are indepedent of one another.

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

- We do this so that after training our model, we can test it on data it hasn't seen. Otherwise, we are training our model just to the specific data we have and have no way of knowing it will be accurate on new data.

Task 2: Data Prep

```
lapply(c("tidyverse", "tidymodels", "caret", "yardstick"), library, character.only = TRUE)
```

```
[[1]]
 [1] "lubridate" "forcats"   "stringr"   "dplyr"     "purrr"     "readr"
 [7] "tidyr"     "tibble"    "ggplot2"   "tidyverse" "stats"     "graphics"
[13] "grDevices" "utils"     "datasets"  "methods"   "base"

[[2]]
 [1] "yardstick"    "workflowsets" "workflows"    "tune"         "rsample"
 [6] "recipes"      "parsnip"      "modeldata"    "infer"        "dials"
[11] "scales"       "broom"        "tidymodels"   "lubridate"    "forcats"
[16] "stringr"      "dplyr"        "purrr"        "readr"        "tidyr"
[21] "tibble"       "ggplot2"      "tidyverse"    "stats"        "graphics"
[26] "grDevices"    "utils"        "datasets"     "methods"      "base"

[[3]]
 [1] "caret"        "lattice"      "yardstick"    "workflowsets" "workflows"
 [6] "tune"         "rsample"      "recipes"      "parsnip"      "modeldata"
[11] "infer"        "dials"        "scales"       "broom"        "tidymodels"
[16] "lubridate"    "forcats"      "stringr"      "dplyr"        "purrr"
[21] "readr"        "tidyr"        "tibble"       "ggplot2"      "tidyverse"
[26] "stats"        "graphics"     "grDevices"    "utils"        "datasets"
[31] "methods"      "base"

[[4]]
 [1] "caret"        "lattice"      "yardstick"    "workflowsets" "workflows"
 [6] "tune"         "rsample"      "recipes"      "parsnip"      "modeldata"
[11] "infer"        "dials"        "scales"       "broom"        "tidymodels"
[16] "lubridate"    "forcats"      "stringr"      "dplyr"        "purrr"
[21] "readr"        "tidyr"        "tibble"       "ggplot2"      "tidyverse"
[26] "stats"        "graphics"     "grDevices"    "utils"        "datasets"
[31] "methods"      "base"
```

```
heartdata <- read_csv("https://www4.stat.ncsu.edu/~online/datasets/heart.csv") %>%
  as.tibble()
```

```
Warning: `as.tibble()` was deprecated in tibble 2.0.0.
i Please use `as_tibble()` instead.
i The signature and semantics have changed, see `?as_tibble`.
```

```
Rows: 918 Columns: 12
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (5): Sex, ChestPainType, RestingECG, ExerciseAngina, ST_Slope
dbl (7): Age, RestingBP, Cholesterol, FastingBS, MaxHR, Oldpeak, HeartDisease

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

1. Run and report summary() on your data set.

```
summary(heartdata)
```

```
      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. What type of variable is Heart Disease?

Quantitative

b. Does this make sense?

No, this variable is meant to be a binary Y/N, for whether or not Heart Disease is present. This would be categorical.

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.

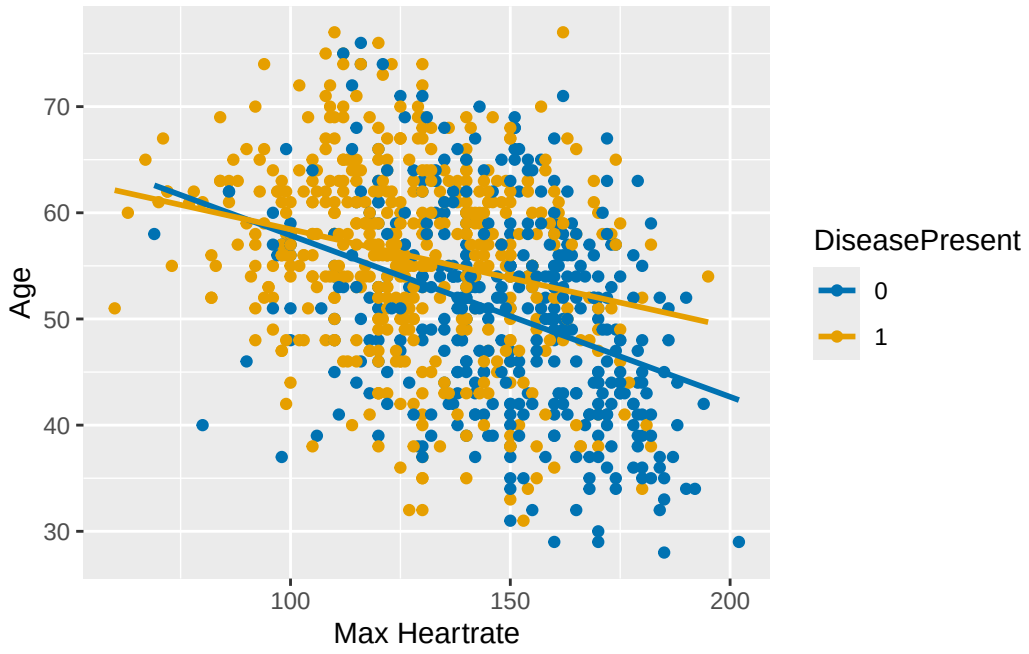
```
new_heart <- heartdata %>% mutate(DiseasePresent = as.factor(HeartDisease),
HeartDisease = NULL, ST_Slope = NULL)
```

Task 3: EDA

1.

Create the appropriate scatterplot to visualize this relationship (age as a function of heart disease and max heart rate)

```
ggplot(new_heart, aes(x = MaxHR, y = Age, color = DiseasePresent)) +
geom_point() +
geom_smooth(method = "lm", se = FALSE) +
labs(x = "Max Heartrate", y= "Age") +
scale_color_manual(values = c("0" = "#0072B2", "1" = "#E69F00"))
```




```
#colors taken from the cbPalette (colorblind friendly Palette) used on the cookbook-R site
```

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

- Looking at the graph, I think that an interaction model is more appropriate, since the lines for DiseasePresent cross. This could indicate that there is a dependence within the variables of the model. We should investigate to see the if impact of MaxHR on Age changes if Disease is Present.

Task 4: Testing and Training

1. Split your data into a training and test set.

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

Task 5: OLS and LASSO

1. 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.

```
ols_mlr <- lm(Age ~ MaxHR + DiseasePresent, data = train)

#Report the summary output.
summary(ols_mlr)
```

```
Call:
lm(formula = Age ~ MaxHR + DiseasePresent, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-21.8743  -5.9127   0.3621   5.9368  21.6244

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     68.49810    2.04318  33.525  < 2e-16 ***
MaxHR           -0.12175    0.01349  -9.022  < 2e-16 ***
DiseasePresent1  3.00330    0.68846   4.362 1.47e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.527 on 731 degrees of freedom
Multiple R-squared:  0.1734,    Adjusted R-squared:  0.1711
F-statistic: 76.65 on 2 and 731 DF,  p-value: < 2.2e-16
```

2. 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.

```
yardstick::rmse_vec(test$Age, predict(ols_mlr, newdata = test))
```

```
[1] 9.006226
```

RMSE = 9.01

3. See if a model fit using LASSO has better predictive performance than with OLS. Use cross validation to select the best tuning parameter, then evaluate our LASSO model on the testing data set and compare RMSEs.

10 fold cv

```
heart_CV_folds <- vfold_cv(train, 10)
```

```
LASSO_recipe <- recipe(Age ~ MaxHR + DiseasePresent, data = train) |>
  step_dummy(DiseasePresent) |>
  step_normalize(all_numeric_predictors()) |>
  step_interact(~MaxHR:starts_with("DiseasePresent"))


LASSO_recipe
```

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



-- Inputs

Number of variables by role

outcome:   1
predictor: 2



-- Operations

* Dummy variables from: DiseasePresent

* Centering and scaling for: all_numeric_predictors()

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

4.

set up spec

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

#add to 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
```

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

#collect the best metric using RMSE
best_met <- LASSO_grid |> select_best(metric = "rmse")
best_met
```

```
# A tibble: 1 x 2
```

```
  penalty .config
    <dbl> <chr>
1  0.0174 Preprocessor1_Model165
```

now we can finalize our workflow by using the lowest RMSE we found to make the best LASSO model

```
LASSO_final <- LASSO_wkf |>
  finalize_workflow(best_met) |>
  fit(train)
tidy(LASSO_final)
```

```
# A tibble: 4 x 3
  term                       estimate penalty
  <chr>                         <dbl>   <dbl>
1 (Intercept)                   54.0   0.0174
2 MaxHR                         -3.08  0.0174
3 DiseasePresent_X1              1.36  0.0174
4 MaxHR_x_DiseasePresent_X1      1.03  0.0174
```

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 them to be roughly the same because the tuning parameter we found to be best for LASSO is so small

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

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

```
[1] 9.095981
```

Though slightly larger, RMSE is also about 9.01 for LASSO

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

Because RMSE is a measure of the quality of predictions made by the model, it isn't taking coefficients into account

Task 6: Logistic Regression

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

```
model1 <- recipe(DiseasePresent ~ MaxHR + ChestPainType + Cholesterol, data = train) |>
step_normalize(all_numeric_predictors())

model2 <- recipe(DiseasePresent ~ MaxHR + Age + RestingECG, data = train) |>
step_normalize(all_numeric_predictors())
```

set up engines for the Log Reg

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

#then the workflows
LR_wkf1 <- workflow() |>
add_recipe(model1) |>
add_model(LR_spec)

LR_wkf2 <- workflow() |>
add_recipe(model2) |>
add_model(LR_spec)
```

use CV to determine best model of the two

```
LR_fit1 <- LR_wkf1 |>
fit_resamples(heart_CV_folds, metrics = metric_set(accuracy, mn_log_loss))
LR_fit2 <- LR_wkf2 |>
fit_resamples(heart_CV_folds, metrics = metric_set(accuracy, mn_log_loss))
```

see which model did the best by collecting the metrics

```
rbind(LR_fit1 |> collect_metrics(),
LR_fit2 |> 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     0.782    10 0.00969 Preprocessor1_Model1
2 Model1 mn_log_loss binary     0.498    10 0.0119  Preprocessor1_Model1
3 Model2 accuracy    binary     0.679    10 0.0181  Preprocessor1_Model1
4 Model2 mn_log_loss binary     0.598    10 0.0127  Preprocessor1_Model1
```

We can see that the first model, with MaxHR, ChestPainType, and Cholesterol, has both higher accuracy in prediction as well as lower log loss, meaning we can have more confidence in it's predictions. This implies that Model1 is the better prediction model for the presence of Heart Disease.

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

```
#fit best model onto train data
LR_train_fit <- LR_wkf1 |> fit(train)

conf_mat(test |> mutate(estimate = LR_train_fit |> predict(test) |> pull()),
DiseasePresent, estimate)
```

```
          Truth
Prediction  0  1
         0 68 18
         1 26 72
```

So we see that the model: correctly predicted the absence of disease 68 times correctly predicted the presence of disease 72 times

incorrectly predicted the absence of disease when it was actually there 18 times incorrectly predicted the presence of disease when there was none 26 times

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

Sensitivity is correctly predicting the presence of disease when there is actually disease = 72 / (72 + 18) = 0.8 or 80% 80% of the time the model correctly predicts the presence of disease

Specificity is correctly predicting the absence of disease = 68 / (68 + 26) = .723 72.3% of the time the model correctly predicts the absence of disease