Modelling

Task 1:Conceptual Questions

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

The purpose of cross-validation when fitting a random forest model is to evalute and validate a model in an unbaised manner. Specifically, in a random forest model, it can be used to help prevent overfitting and also help tune parameters.

- Describe the bagged tree algorithm.

The bagged tree algorithm creates many bootstrap resamples and fits trees to each re sample. For the set of given predictors, it finds the predicted value for each tree. Then you combine the predictions from the trees to create a final predictions. For regression models, usually you use the average of the predictions. For the classication tree you use the most common prediction made by all the trees.

- What is meant by a general linear model?

A general linear model is the linear regression model with a continuous and normally distributed response variable and predictors that can be continous or categorical. This can include both single and multiple regression models.

- 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?

The interaction term allows for the effect of one variables to depend on the value of another. When you just include the main effect in a model, you are only looking at the individual effect of each variable. If you have an interaction term, you can look at the effect of the interaction between both variables.

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

The reason we split our data into a training and test set is to determine if our model can predict observations well it has yet to see. This will let us avoid overfitting the model.

Task 2: Data Prep

###Packages and daata

```r
library(tidyverse)
library(tidymodels)
library(yardstick)
library(ggplot2)
library(glmnet)

#Read in data as tibble
df_heart <- read.csv("heart.csv")
df_heart <- as_tibble(df_heart)

#Summarize data
summary(df_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
```

1. Run and report summary() on your data set. Then, answer the following questions:

- What type of variable (in R) is Heart Disease? Categorical or Quantitative? It is currently a quantitative variable in R.

- Does this make sense? Why or why not.

This does not make sense. It should be a categorical variable since the only values are 1 and 0.

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. We will use this new data set for the remainder of the assignment.

```
new_heart <- df_heart |>
  mutate(HeartDisease = factor(HeartDisease)) |>
 select(-ST_Slope)

summary(new_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        HeartDisease
 Length:918         Min.   :-2.6000   0:410       
 Class :character   1st Qu.: 0.0000   1:508       
 Mode  :character   Median : 0.6000               
                    Mean   : 0.8874               
                    3rd Qu.: 1.5000               
                    Max.   : 6.2000               
```

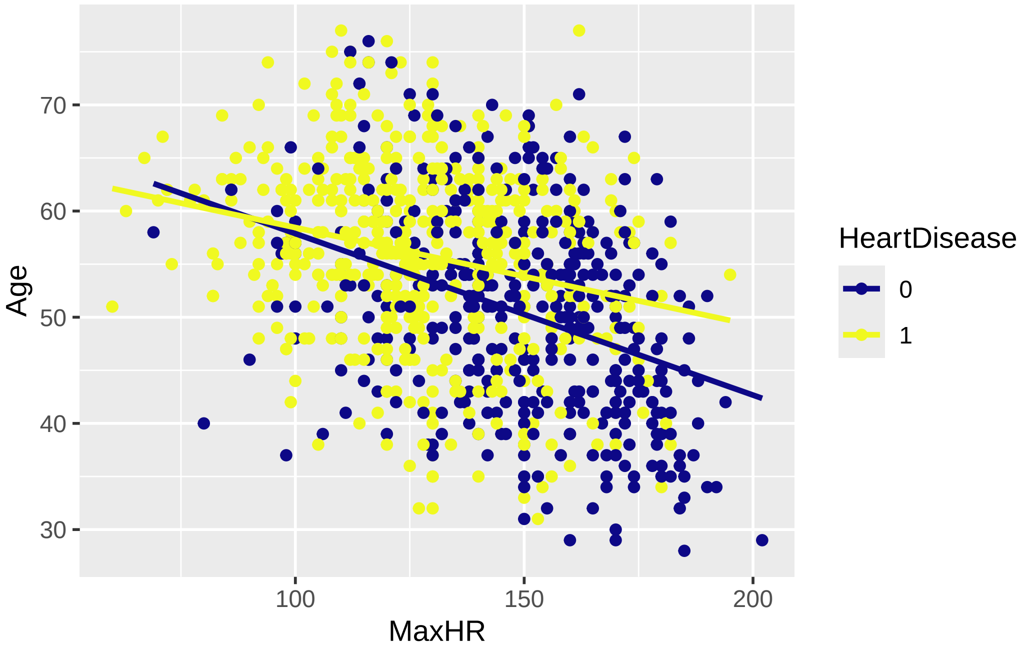
##Task 3: EDA

1. We are going to model someone's age (our response variable) as a function of heart disease and their max heart rate. First, create the appropriate scatterplot to visualize

this relationship. Add a line to the scatterplot that represents if someone has or does not have heart disease. Remove the standard error bars from the lines and add appropriate labels. Also, change the color pallet to be more colorblind friendly.

```r
ggplot(data = new_heart, aes(x = MaxHR, y = Age, color =HeartDisease)) +
 geom_point() +
  geom_smooth(method ="lm",se=FALSE) +
  scale_color_viridis_d(option ="C")
```

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



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

Based on the visual evidence, this looks like it will be an interaction model. There seems to be an interaction between MaxHR and heart disease as those with lower MaxHR tend not to have heart disease and those with higher MaxHR do have heart disease.

##Task 4: Testing and Training

```r
set.seed(101)
heart_split <- initial_split(new_heart,prop=.8)
test <- testing(heart_split)
train <- training(heart_split)
```

##Task 5: OLS LASSO

1. Regardless of your answer in Task 3, we are going to fit an interaction model. First 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. Report the summary output.

```
ols_mlr <- lm(Age ~ MaxHR * HeartDisease, data=train)

summary(ols_mlr)
```

```
Call:
lm(formula = Age ~ MaxHR * HeartDisease, 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 ***
HeartDisease1       -8.58502    3.83433  -2.239  0.02546 *  
MaxHR:HeartDisease1  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
```

2. We are going to 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.

```
#Use predict on new model
test |>
  select(Age,MaxHR, HeartDisease)|>
  mutate(model_1 = predict(ols_mlr, newdata=test))
```

```
# A tibble: 184 x 4
     Age MaxHR HeartDisease model_1
   <int> <int> <fct>          <dbl>
 1    54   142 0               51.5
 2    48   120 0               55.2
 3    58    99 1               58.4
 4    54   150 0               50.1
 5    53   127 0               54.0
 6    43   154 0               49.4
 7    54   100 1               58.4
 8    52   170 0               46.7
 9    43   120 1               56.6
10    37   168 0               47.0
# i 174 more rows
```

```r
#obtain the test set RMSE
rmse_vec(test$Age,predict(ols_mlr, newdata=test))
```

```
[1] 9.100206
```

3. Now, we are going to see if a model fit using LASSO has better predictive performance than with OLS. We are going to use cross validation to select the best tuning parameter, and then evaluate our LASSO model on the testing data set and compare RMSEs.

```r
#Create fold VC to choose our tuning parameter
heart_CV_folds <- vfold_cv(train, 10)

#Set up LASSO_recipe
#use a + to separate your explanatory variables, regardless ofthe
#fact that this will be an interaction model
#Standardize your variables as normal
#Add an additional pipe into the function step_interact().
#Within step_interact(), start with a ~, and then put
#variable_name_1:starts_with("variable_name_2_

LASSO_recipe <- recipe(Age ~ MaxHR + HeartDisease, data=train) |>
  step_dummy(HeartDisease)|>
  step_normalize(all_predictors()) |>
  step_interact(~ MaxHR:starts_with("HeartDisease_"))

LASSO_recipe
```

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



-- Inputs

Number of variables by role

outcome:   1
predictor: 2



-- Operations

* Dummy variables from: HeartDisease

* Centering and scaling for: all_predictors()

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

4. Now, set up your appropriate spec, and grid. Next, select your final model, and report the results using the tidy() function around your model name.

```r
#Set up your appropriate spec

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

#Set up 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
```

```r
#Set up appropriate grid
LASSO_grid <- LASSO_wkf |>
  tune_grid(resamples = heart_CV_folds ,
            grid = grid_regular(penalty(), levels = 200))

LASSO_grid
```

```
# Tuning results
# 10-fold cross-validation
# A tibble: 10 x 4
   splits            id     .metrics             .notes
   <list>            <chr>  <list>               <list>
 1 <split [660/74]> Fold01 <tibble [400 x 5]> <tibble [0 x 3]>
 2 <split [660/74]> Fold02 <tibble [400 x 5]> <tibble [0 x 3]>
 3 <split [660/74]> Fold03 <tibble [400 x 5]> <tibble [0 x 3]>
 4 <split [660/74]> Fold04 <tibble [400 x 5]> <tibble [0 x 3]>
 5 <split [661/73]> Fold05 <tibble [400 x 5]> <tibble [0 x 3]>
 6 <split [661/73]> Fold06 <tibble [400 x 5]> <tibble [0 x 3]>
 7 <split [661/73]> Fold07 <tibble [400 x 5]> <tibble [0 x 3]>
 8 <split [661/73]> Fold08 <tibble [400 x 5]> <tibble [0 x 3]>
 9 <split [661/73]> Fold09 <tibble [400 x 5]> <tibble [0 x 3]>
10 <split [661/73]> Fold10 <tibble [400 x 5]> <tibble [0 x 3]>
```

```
#Select your final model and report the results using
#the tidy() function around your model name

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

```
# A tibble: 1 x 2
  penalty .config
    <dbl> <chr>
1  0.0174 Preprocessor1_Model165
```

```
LASSO_wkf |>
  finalize_workflow(lowest_rmse)
```

```
== 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 = 0.0174263338600965
  mixture = 1

Computational engine: glmnet
```

```
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)                 54.0   0.0174
2 MaxHR                       -3.08  0.0174
3 HeartDisease_X1              1.36  0.0174
4 MaxHR_x_HeartDisease_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 woudl expect the RMSE calculations to be different given the difference in coefficients.

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

```r
#obtain the test set RMSE
rmse_vec(test$Age,predict(ols_mlr, newdata=test))
```

```
[1] 9.100206
```

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

```
[1] 9.095981
```

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

The reason that the RMSE calculations are roughly the same because the LASSO model introduces penalties so the coefficients are able to be minimized and produce simpler models. It makes it simpler by minimizing irrelevant coefficents to 0. The linear least squares just minimizes error without taking into account the models complexity. So while the least squares does minimize error, it does not look at various values of coefficients.

##Task 6: Logistic Regression

Propose two different logistic regression models with heart disease as our response. Note: You don't have to use the dummy columns you made here as the glm() function (and the caret implementation of it) can handle factor/character variables as predictors.

```
#Propose two different logistic regression models with Heart Disease as our response

summary(new_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       HeartDisease
 Length:918         Min.   :-2.6000   0:410       
 Class :character   1st Qu.: 0.0000   1:508       
 Mode  :character   Median : 0.6000               
                    Mean   : 0.8874               
                    3rd Qu.: 1.5000               
                    Max.   : 6.2000               
```

```
new_heart |>
 group_by(HeartDisease) |>
 summarize(mean_Age = mean(Age),
 mean_RestingBP = mean(RestingBP))
```

```
# A tibble: 2 x 3
  HeartDisease mean_Age mean_RestingBP
  <fct>           <dbl>          <dbl>
1 0                50.6           130.
2 1                55.9           134.
```

```
#One model with gender and cholesterol
```

```r
new_heart |>
 group_by(HeartDisease,Sex) |>
 summarize(Cholesterol = mean(Cholesterol))
```

```
`summarise()` has grouped output by 'HeartDisease'. You can override using the
`.groups` argument.
```

```
# A tibble: 4 x 3
# Groups:   HeartDisease [2]
  HeartDisease Sex   Cholesterol
  <fct>        <chr>       <dbl>
1 0            F            247.
2 0            M            216.
3 1            F            223.
4 1            M            171.
```

```r
LR1_rec <- recipe(HeartDisease ~ Cholesterol + Sex,
 data = train) |>
 step_normalize(Cholesterol) |>
 step_dummy(Sex)

#One model with gender and MaxHR

new_heart |>
 group_by(HeartDisease,Sex) |>
 summarize(MaxHR = mean(MaxHR))
```

```
`summarise()` has grouped output by 'HeartDisease'. You can override using the
`.groups` argument.
```

```
# A tibble: 4 x 3
# Groups:   HeartDisease [2]
  HeartDisease Sex   MaxHR
  <fct>        <chr> <dbl>
1 0            F      149.
2 0            M      148.
3 1            F      138.
4 1            M      127.
```

```
LR2_rec <- recipe(HeartDisease ~ Sex + MaxHR,
 data = train) |>
 step_normalize(MaxHR) |>
 step_dummy(Sex)
```

Fit those models on the training set, using repeated CV.

```
#Fit model on the training set using repeated CV

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))
```

Identify your best performing model. Justify why this is your best performing model and provide a basic summary of it.

```
#Identify your best performing model

rbind(LR1_fit |> collect_metrics(),
 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     0.651    10 0.0169  Preprocessor1_Model1
2 Model1 mn_log_loss binary     0.625    10 0.00992 Preprocessor1_Model1
```

```
3 Model2 accuracy    binary     0.699    10 0.0143  Preprocessor1_Model1
4 Model2 mn_log_loss binary     0.579    10 0.0121  Preprocessor1_Model1
```

```r
#Compare to the proportions of 1's in the training data
mean(train$HeartDisease == "1")
```

```
[1] 0.5694823
```

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

```r
#Identify your best performing model

LR_train_fit <- LR2_wkf |>
 fit(train)

conf_mat(train |> mutate(estimate = LR_train_fit |> predict(train) |> pull()), #data
 HeartDisease, #truth
 estimate) #estimate from model
```

```
          Truth
Prediction   0   1
         0 181  83
         1 135 335
```

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

```r
#Identify the values of sensitivity and specificity, and interpret them in the context of th

#Sensitivity
TP <- 335
FP <- 135
FN <-83
TN <- 181

sensitivity <- TP / (TP+FN)
sensitivity
```

```
[1] 0.8014354
```

```
#Specificity

specificity <- TN/(TN+FP)
specificity
```

```
[1] 0.5727848
```

This model has high sensitivity (80%), meaning it identifies true positives very well. The specificity, the measurement of identifing ture negatives, is not as good as 57%.