SYS 6108 Final Project: Unveiling Heart Attack Risk Through Data Insights

AUTHOR
Michael R. Burns and A. James Caldwell

PUBLISHED
May 8, 2024

Unveiling Heart Attack Risk Through Data Insights

Link to data: https://www.kaggle.com/competitions/heart-attack-risk-analysis

```r
# Initialize the required R packages and directories {.unnumbered .unlisted}
data_dir = 'https://mdporter.github.io/SYS6018/data/' # data directory
library(dplyr)
library(mclust)    # for model-based clustering
library(mixtools)  # for poisson mixture mode
library(tidyverse) # functions for data manipulation
library(ranger)       # fast random forest implementation
library(glmnet)    # for glmnet() functions
library(yardstick) # for evaluation metrics
library(xgboost)
library(caret)
```

The goal of this project is to predict whether a patient is at a high or low risk of a heart attack. We are using the Kaggle dataset “Heart Attack Risk Analysis” which has medical and demographic information for over 7000 patients. The heart attack risk variable is the outcome and all others are predictors.

Data Loading

Set the Seed

```r
set.seed(227)
```

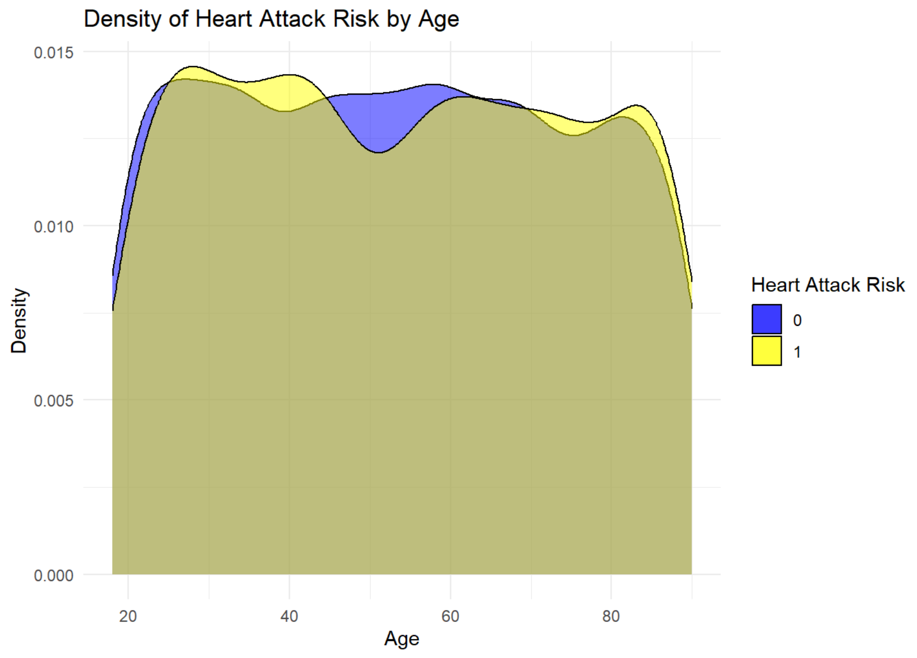
Load the Data set

```
fp_train <- 'C:/Users/michael.burns/OneDrive - University of Virginia/2024 Spring Clas
fp_test <- 'C:/Users/michael.burns/OneDrive - University of Virginia/2024 Spring Class

# fp_train <- 'C:/Users/james.caldwell/OneDrive - University of Virginia/Documents/4Sc
# fp_test <- 'C:/Users/james.caldwell/OneDrive - University of Virginia/Documents/4Sch

# Load the data
train = read.csv(fp_train)
test = read.csv(fp_test)
```

Data Exploration

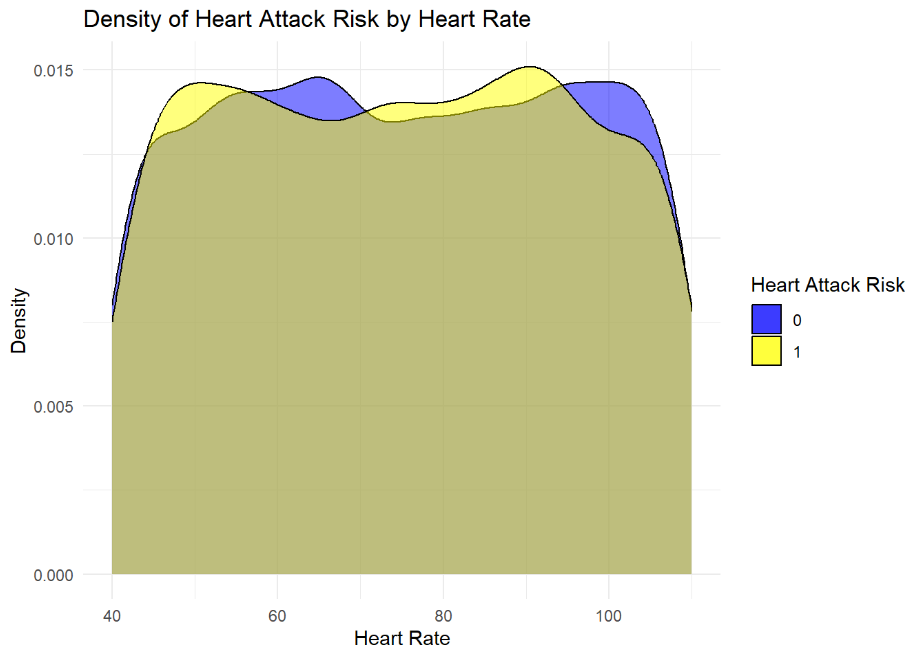
Before modeling, we did some brief data exploration. Here we show the density/percentages of patient correlation of BMI, heart rate, age, and previous heart problems to their heart attack risk level.

```
# Filter data for Heart.Attack.Risk = 0 and 1
risk_0 <- subset(train, Heart.Attack.Risk == 0)
risk_1 <- subset(train, Heart.Attack.Risk == 1)

# Create density plots
plot_age <- ggplot(train, aes(x = Age)) +
  geom_density(data = risk_0, aes(fill = "0"), alpha = 0.5) +
  geom_density(data = risk_1, aes(fill = "1"), alpha = 0.5) +
  labs(x = "Age", y = "Density", title = "Density of Heart Attack Risk by Age") +
  theme_minimal() +
  scale_fill_manual(name = "Heart Attack Risk", values = c("0" = "blue", "1" = "yellow
# Display the plot
print(plot_age)
```



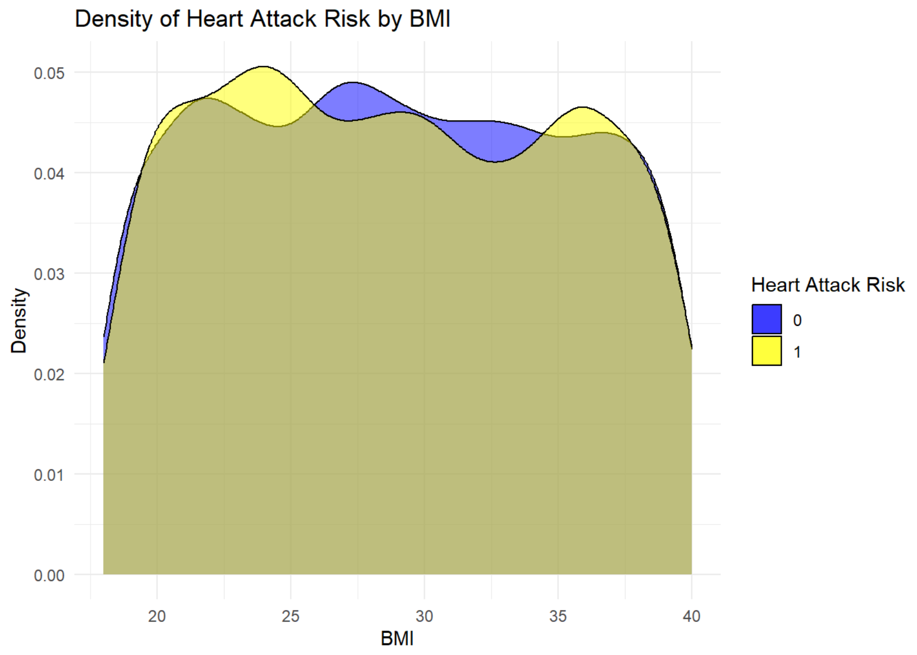
The density of observations is plotted for age here broken down by those with a heart attack risk of 0 and 1. From this we can see that there are minimal differences between the ages in the two risk categories.

```
        # Create density plots
        plot_HR <- ggplot(train, aes(x = Heart.Rate)) +
          geom_density(data = train[train$Heart.Attack.Risk == 0,], aes(fill = "0"), alpha = 0
          geom_density(data = train[train$Heart.Attack.Risk == 1,], aes(fill = "1"), alpha = 0
          labs(x = "Heart Rate", y = "Density", title = "Density of Heart Attack Risk by Heart
          theme_minimal() +
          scale_fill_manual(name = "Heart Attack Risk", values = c("0" = "blue", "1" = "yellow
        # Display the plot
        print(plot_HR)
```



The density of observations is plotted for heart rate here broken down by those with a heart attack risk of 0 and 1. From this we can see that there are minimal differences between the heart rates in the two risk categories.

```
# Create density plots
plot_BMI  <- ggplot(train, aes(x = BMI)) +
  geom_density(data = train[train$Heart.Attack.Risk == 0,], aes(fill = "0"), alpha = 0
  geom_density(data = train[train$Heart.Attack.Risk == 1,], aes(fill = "1"), alpha = 0
  labs(x = "BMI", y = "Density", title = "Density of Heart Attack Risk by BMI") +
  theme_minimal() +
  scale_fill_manual(name = "Heart Attack Risk", values = c("0" = "blue", "1" = "yellow
# Display the plot
print(plot_BMI )
```



The density of observations is plotted for BMI broken down by those with a heart attack risk of 0 and 1. From this we can see that there are minimal differences between the BMI in the two risk categories.From these plots and others generated we can tell that there are minimal (if any) obvious differences in the populations between those with and without heart attack risk.

We then wanted to look at the correlation of a high heart attack risk for those with previous heart problems. We generated a risk percentage of those with and without previous problems (given the state of previous problems what is the percentage that have a heart attack risk of 1). This code could be used to look at any other binary predictor variables as well.

```
# Calculate the total number of individuals with and without previous heart problems
total_with_previous_heart_problems <- sum(train$Previous.Heart.Problems == 1)
total_without_previous_heart_problems <- sum(train$Previous.Heart.Problems == 0)

# Calculate the number of individuals with and without previous heart problems who hav
with_previous_heart_attack <- sum(train$Previous.Heart.Problems == 1 & train$Heart.Att
without_previous_heart_attack <- sum(train$Previous.Heart.Problems == 0 & train$Heart.

# Calculate the percentage of individuals with and without previous heart problems who
percentage_with_previous_heart_attack <- (with_previous_heart_attack / total_with_prev
percentage_without_previous_heart_attack <- (without_previous_heart_attack / total_wit
```

```r
# Print the percentages
cat("Percentage of individuals with previous heart problems and heart attack risk:", p
```

```
Percentage of individuals with previous heart problems and heart attack risk: 36.4 %
```

```r
cat("Percentage of individuals without previous heart problems and heart attack risk:"
```

```
Percentage of individuals without previous heart problems and heart attack risk: 35.1 %
```

Here we can see that there is little difference in the heart attack risk between those with and without previous heart problems since the risk for those with is ~1% higher. This is going to come into play in our analysis since there are only slight differences in any given variable that will lead to a different risk outcome.

Summarizing the data exploration

The data is complex with multiple interaction points for the numeric data. For the binary data, there are small differences between the outputs for risk level predictors. Any model that will predict well for this dataset will have to carefully balance high complexity while avoiding overfitting.

Clean the data

Many steps needed to be taken to take the data from its raw state to something that could be used by data mining methods.

Patient ID

Patient ID was a unique value for each observation and thus carried no modeling utility so it was removed from the data set.

```r
#first need to remove patient ID from both data sets because it is unique so its not h
data_train <- subset(train, select = -Patient.ID)
data_test <- subset(test, select = -Patient.ID)
```

Blood Pressure

Blood pressure is reported as two numbers. The value is commonly reported as Systolic/Diastolic (https://www.heart.org/en/health-topics/high-blood-pressure/understanding-blood-pressure-readings) where the values are reported together. Since each of these values means something about the patient and the models would not be able to handle the slash, this was separated into two different variables. The upper refers to the systolic pressure and the lower refers to the diastolic pressure.

```
#Handle Blood Pressure
#On the training data
data_train <- separate(data_train, Blood.Pressure, into = c("Blood.Pressure.Upper","Bl
# Convert the newly created columns to numeric
data_train$Blood.Pressure.Upper <- as.numeric(data_train$Blood.Pressure.Upper)
data_train$Blood.Pressure.Lower <- as.numeric(data_train$Blood.Pressure.Lower)
#On the test data
data_test <- separate(data_test, Blood.Pressure, into = c("Blood.Pressure.Upper","Bloo
# Convert the newly created columns to numeric
data_test$Blood.Pressure.Upper <- as.numeric(data_test$Blood.Pressure.Upper)
data_test$Blood.Pressure.Lower <- as.numeric(data_test$Blood.Pressure.Lower)
```

One-hot Encoding and Scaling

All of the data needed additional cleaning. For the categorical variables we needed to use one-hot encoding for the numerical models to be able to handle the data. The one-hot encoding split each of the categorical variables into a different column for each entry. Since we had measurements with different magnitudes we also needed to scale our numerical data. We scaled all of the numeric variables to have a mean of 0 and a standard deviation of 1. This brought all of our values to a level where they could be put into the same model and one would not artificially outweigh another.

```
#First split off the outcome from the training data
data_train_out <- subset(data_train, select=Heart.Attack.Risk)
data_train_input <- subset(data_train, select=-Heart.Attack.Risk)
#combine the train and test data for consistency in preprocessing
combined_data <- rbind(data_train_input,data_test)

#while in this state I need to scale all of our data
```

```
numeric_column_names <- c("Age","Cholesterol","Blood.Pressure.Upper","Blood.Pressure.L
combined_data[,numeric_column_names] <- scale(combined_data[,numeric_column_names])

#use makeX to handle 1 hot encoding
combined_data_processed <- makeX(combined_data)
#now separate back into train and test again
data_train <- combined_data_processed[1:nrow(data_train),]
data_test <- combined_data_processed[(nrow(data_train) + 1):nrow(combined_data),]

#convert back to a data frame
data_train <- as.data.frame(data_train)
data_test <- as.data.frame(data_test)

#now add the outcome column back into the training data
data_train <- cbind(data_train, Heart.Attack.Risk = data_train_out)

#replace all of the "." and " " with "_"
names(data_train) <- gsub("[. ]","_", names(data_train))
names(data_test) <- gsub("[. ]","_", names(data_test))
```

Description of our data set

Our data consists of:

```
#predictors count
num_predictors <- ncol(train) - 2
print(paste("There are", num_predictors,"predictor variables in the raw data."))
```

```
[1] "There are 24 predictor variables in the raw data."
```

```
#observations
num_observations <- nrow(train)
print(paste("There are", num_observations,"observations in the data set."))
```

```
[1] "There are 7010 observations in the data set."
```

```
#predictors count
num_predictors <- ncol(data_train) - 1
```

```
print(paste("After the data processing there are", num_predictors,"predictor variables
```

```
[1] "After the data processing there are 53 predictor variables for the data."
```

The number of observations is unchanged by the data processing.

The following is a summary of the variables after one hot encoding and scaling are done.

```
summary(data_train)
```

```
      Age            SexFemale         SexMale         Cholesterol
 Min.   :-1.680   Min.   :0.000   Min.   :0.000   Min.   :-1.730
 1st Qu.:-0.880   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:-0.839
 Median :-0.033   Median :0.000   Median :1.000   Median :-0.011
 Mean   :-0.009   Mean   :0.302   Mean   :0.698   Mean   : 0.000
 3rd Qu.: 0.861   3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.: 0.855
 Max.   : 1.708   Max.   :1.000   Max.   :1.000   Max.   : 1.733
 Blood_Pressure_Upper Blood_Pressure_Lower   Heart_Rate         Diabetes
 Min.   :-1.711       Min.   :-1.714       Min.   :-1.704   Min.   :0.000
 1st Qu.:-0.876       1st Qu.:-0.896       1st Qu.:-0.877   1st Qu.:0.000
 Median :-0.003       Median :-0.011       Median :-0.001   Median :1.000
 Mean   :-0.002       Mean   : 0.000       Mean   : 0.004   Mean   :0.653
 3rd Qu.: 0.870       3rd Qu.: 0.875       3rd Qu.: 0.875   3rd Qu.:1.000
 Max.   : 1.705       Max.   : 1.693       Max.   : 1.702   Max.   :1.000
 Family_History      Smoking          Obesity     Alcohol_Consumption
 Min.   :0.000   Min.   :0.000   Min.   :0.0   Min.   :0.000
 1st Qu.:0.000   1st Qu.:1.000   1st Qu.:0.0   1st Qu.:0.000
 Median :0.000   Median :1.000   Median :0.0   Median :1.000
 Mean   :0.492   Mean   :0.896   Mean   :0.5   Mean   :0.596
 3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:1.0   3rd Qu.:1.000
 Max.   :1.000   Max.   :1.000   Max.   :1.0   Max.   :1.000
 Exercise_Hours_Per_Week  DietAverage      DietHealthy     DietUnhealthy
 Min.   :-1.731          Min.   :0.000   Min.   :0.000   Min.   :0.000
 1st Qu.:-0.859          1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000
 Median :-0.005          Median :0.000   Median :0.000   Median :0.000
 Mean   :-0.006          Mean   :0.334   Mean   :0.335   Mean   :0.331
 3rd Qu.: 0.867          3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:1.000
 Max.   : 1.726          Max.   :1.000   Max.   :1.000   Max.   :1.000
 Previous_Heart_Problems Medication_Use  Stress_Level
 Min.   :0.000           Min.   :0.0    Min.   :-1.563
 1st Qu.:0.000           1st Qu.:0.0    1st Qu.:-0.864
```

```
Median :0.000          Median :1.0    Median :-0.164
Mean   :0.498          Mean   :0.5    Mean   :-0.006
3rd Qu.:1.000          3rd Qu.:1.0    3rd Qu.: 0.885
Max.   :1.000          Max.   :1.0    Max.   : 1.584
Sedentary_Hours_Per_Day     Income            BMI         Triglycerides
Min.   :-1.729          Min.   :-1.715   Min.   :-1.723   Min.   :-1.733
1st Qu.:-0.872          1st Qu.:-0.867   1st Qu.:-0.865   1st Qu.:-0.879
Median :-0.016          Median :-0.011   Median :-0.024   Median :-0.007
Mean   : 0.000          Mean   : 0.000   Mean   :-0.002   Mean   :-0.004
3rd Qu.: 0.872          3rd Qu.: 0.856   3rd Qu.: 0.859   3rd Qu.: 0.873
Max.   : 1.733          Max.   : 1.758   Max.   : 1.757   Max.   : 1.709
Physical_Activity_Days_Per_Week Sleep_Hours_Per_Day CountryArgentina
Min.   :-1.529                  Min.   :-1.521      Min.   :0.000
1st Qu.:-0.653                  1st Qu.:-1.018      1st Qu.:0.000
Median :-0.215                  Median :-0.012      Median :0.000
Mean   : 0.001                  Mean   : 0.001      Mean   :0.055
3rd Qu.: 0.662                  3rd Qu.: 0.994      3rd Qu.:0.000
Max.   : 1.538                  Max.   : 1.497      Max.   :1.000
CountryAustralia CountryBrazil   CountryCanada   CountryChina  CountryColombia
Min.   :0.000    Min.   :0.000   Min.   :0.00   Min.   :0.00   Min.   :0.000
1st Qu.:0.000    1st Qu.:0.000   1st Qu.:0.00   1st Qu.:0.00   1st Qu.:0.000
Median :0.000    Median :0.000   Median :0.00   Median :0.00   Median :0.000
Mean   :0.052    Mean   :0.053   Mean   :0.05   Mean   :0.05   Mean   :0.049
3rd Qu.:0.000    3rd Qu.:0.000   3rd Qu.:0.00   3rd Qu.:0.00   3rd Qu.:0.000
Max.   :1.000    Max.   :1.000   Max.   :1.00   Max.   :1.00   Max.   :1.000
CountryFrance   CountryGermany   CountryIndia    CountryItaly
Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000
1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000
Median :0.000   Median :0.000   Median :0.000   Median :0.000
Mean   :0.052   Mean   :0.055   Mean   :0.045   Mean   :0.048
3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:0.000
Max.   :1.000   Max.   :1.000   Max.   :1.000   Max.   :1.000
 CountryJapan   CountryNew_Zealand CountryNigeria  CountrySouth_Africa
Min.   :0.000   Min.   :0.00       Min.   :0.000   Min.   :0.000
1st Qu.:0.000   1st Qu.:0.00       1st Qu.:0.000   1st Qu.:0.000
Median :0.000   Median :0.00       Median :0.000   Median :0.000
Mean   :0.049   Mean   :0.05       Mean   :0.051   Mean   :0.049
3rd Qu.:0.000   3rd Qu.:0.00       3rd Qu.:0.000   3rd Qu.:0.000
Max.   :1.000   Max.   :1.00       Max.   :1.000   Max.   :1.000
CountrySouth_Korea  CountrySpain   CountryThailand CountryUnited_Kingdom
Min.   :0.000      Min.   :0.000   Min.   :0.000   Min.   :0.000
1st Qu.:0.000      1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000
```

```
Median :0.000      Median :0.000    Median :0.000    Median :0.000
Mean   :0.047      Mean   :0.048    Mean   :0.049    Mean   :0.052
3rd Qu.:0.000      3rd Qu.:0.000    3rd Qu.:0.000    3rd Qu.:0.000
Max.   :1.000      Max.   :1.000    Max.   :1.000    Max.   :1.000
CountryUnited_States CountryVietnam ContinentAfrica ContinentAsia
Min.   :0.000        Min.   :0.00   Min.   :0.0     Min.   :0.000
1st Qu.:0.000        1st Qu.:0.00   1st Qu.:0.0     1st Qu.:0.000
Median :0.000        Median :0.00   Median :0.0     Median :0.000
Mean   :0.047        Mean   :0.05   Mean   :0.1     Mean   :0.289
3rd Qu.:0.000        3rd Qu.:0.00   3rd Qu.:0.0     3rd Qu.:1.000
Max.   :1.000        Max.   :1.00   Max.   :1.0     Max.   :1.000
ContinentAustralia ContinentEurope ContinentNorth_America
Min.   :0.000      Min.   :0.000   Min.   :0.000
1st Qu.:0.000      1st Qu.:0.000   1st Qu.:0.000
Median :0.000      Median :0.000   Median :0.000
Mean   :0.102      Mean   :0.255   Mean   :0.097
3rd Qu.:0.000      3rd Qu.:1.000   3rd Qu.:0.000
Max.   :1.000      Max.   :1.000   Max.   :1.000
ContinentSouth_America HemisphereNorthern_Hemisphere
Min.   :0.000          Min.   :0.000
1st Qu.:0.000          1st Qu.:0.000
Median :0.000          Median :1.000
Mean   :0.157          Mean   :0.646
3rd Qu.:0.000          3rd Qu.:1.000
Max.   :1.000          Max.   :1.000
HemisphereSouthern_Hemisphere Heart_Attack_Risk
Min.   :0.000                 Min.   :0.000
1st Qu.:0.000                 1st Qu.:0.000
Median :0.000                 Median :0.000
Mean   :0.354                 Mean   :0.357
3rd Qu.:1.000                 3rd Qu.:1.000
Max.   :1.000                 Max.   :1.000
```

```r
num_pos <- sum(data_train$Heart_Attack_Risk == 1)
num_neg <- sum(data_train$Heart_Attack_Risk == 0)
print(paste("There are", num_pos,"observations with a heart attack risk of 1"))
```

```
[1] "There are 2504 observations with a heart attack risk of 1"
```

```r
print(paste("There are", num_neg,"observations with a heart attack risk of 0"))
```

```
[1] "There are 4506 observations with a heart attack risk of 0"
```

Holdout and Test Sets

We created a holdout set of the data from the total training data set. The holdout was sampled randomly as 1/10 of the data set. The holdout data set was created for evaluation of the models so that we can establish a metric for assessment. 1/10 of the data was chosen to make each set large enough for their purposes.

```
#get the holdout data
# Step 1: Determine the total number of rows
total_rows <- nrow(data_train)

# Step 2: Calculate the number of rows for holdout (one-tenth of total)
holdout_size <- round(total_rows / 10)

# Step 3: Randomly select row indices for holdout
holdout_indices <- sample(1:total_rows, holdout_size)

# Step 4: Create the holdout dataset by subsetting the original data table
holdout_data <- data_train[holdout_indices, ]

# Remove the holdout rows from the original dataset to get the training dataset
training_data <- data_train[-holdout_indices, ]
```

The following are the data after this split:

```
#observations in training set
num_train <- nrow(training_data)
print(paste("There are", num_train,"observations in the training data set."))
```

```
[1] "There are 6309 observations in the training data set."
```

```
#observations in holdout set
num_hold <- nrow(holdout_data)
print(paste("There are", num_hold,"observations in the holdout data set."))
```

```
[1] "There are 701 observations in the holdout data set."
```

We needed to check that this random sample did not unevenly select the data with respect to heart attack risk. One way of doing this is to check the data sets to see if their proportions in the response variable agreed.

```
#percentage of observations that had a heart attack in training set
pct_train <- (sum(training_data$Heart_Attack_Risk == 1)/nrow(training_data))*100
print(paste(pct_train,"% of the observations in the training data set had a positive h
```

```
[1] "35.8535425582501 % of the observations in the training data set had a positive heart
attack risk."
```

```
#percentage of observations that had a heart attack in training set
pct_hold <- (sum(holdout_data$Heart_Attack_Risk == 1)/nrow(holdout_data))*100
print(paste(pct_hold,"% of the observations in the holdout data set had a positive hea
```

```
[1] "34.5221112696148 % of the observations in the holdout data set had a positive heart
attack risk."
```

Since these percentages are close we are okay with the training/holdout split here being considered random.

Get data subsets

For simplicity of model inputs we split the data into the predictor and outcome/output variables. The outcome was the Heart attack risk and all others were predictors

```
#get the subset of holdout data
holdout_predictors <- subset(holdout_data, select = -Heart_Attack_Risk)
holdout_outcome <- holdout_data$Heart_Attack_Risk

#get the subsets of training data
train_predictors <- subset(training_data, select = -Heart_Attack_Risk)
train_output <- training_data$Heart_Attack_Risk
```

Data characteristics of the training data used for model creation

The characteristics of the training data (after the holdout/training split) are shown here. There are no apparent differences between this and the summary before the split, so this process has not changed our sample.

```
summary(training_data)
```

```
      Age             SexFemale          SexMale         Cholesterol     
 Min.   :-1.680   Min.   :0.000   Min.   :0.000   Min.   :-1.730  
 1st Qu.:-0.880   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:-0.839  
 Median :-0.033   Median :0.000   Median :1.000   Median :-0.011  
 Mean   :-0.010   Mean   :0.302   Mean   :0.698   Mean   : 0.000  
 3rd Qu.: 0.861   3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.: 0.855  
 Max.   : 1.708   Max.   :1.000   Max.   :1.000   Max.   : 1.733  
 Blood_Pressure_Upper Blood_Pressure_Lower   Heart_Rate         Diabetes    
 Min.   :-1.711       Min.   :-1.714       Min.   :-1.704   Min.   :0.000  
 1st Qu.:-0.876       1st Qu.:-0.896       1st Qu.:-0.877   1st Qu.:0.000  
 Median :-0.003       Median :-0.011       Median :-0.001   Median :1.000  
 Mean   :-0.004       Mean   : 0.001       Mean   : 0.002   Mean   :0.651  
 3rd Qu.: 0.870       3rd Qu.: 0.875       3rd Qu.: 0.875   3rd Qu.:1.000  
 Max.   : 1.705       Max.   : 1.693       Max.   : 1.702   Max.   :1.000  
 Family_History     Smoking          Obesity       Alcohol_Consumption
 Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000      
 1st Qu.:0.000   1st Qu.:1.000   1st Qu.:0.000   1st Qu.:0.000      
 Median :0.000   Median :1.000   Median :0.000   Median :1.000      
 Mean   :0.494   Mean   :0.897   Mean   :0.497   Mean   :0.597      
 3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:1.000      
 Max.   :1.000   Max.   :1.000   Max.   :1.000   Max.   :1.000      
 Exercise_Hours_Per_Week  DietAverage      DietHealthy     DietUnhealthy  
 Min.   :-1.731          Min.   :0.000   Min.   :0.000   Min.   :0.000  
 1st Qu.:-0.855          1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000  
 Median :-0.006          Median :0.000   Median :0.000   Median :0.000  
 Mean   :-0.007          Mean   :0.336   Mean   :0.332   Mean   :0.331  
 3rd Qu.: 0.863          3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:1.000  
 Max.   : 1.726          Max.   :1.000   Max.   :1.000   Max.   :1.000  
 Previous_Heart_Problems Medication_Use   Stress_Level   
 Min.   :0.000           Min.   :0.000   Min.   :-1.563  
 1st Qu.:0.000           1st Qu.:0.000   1st Qu.:-0.864  
 Median :1.000           Median :1.000   Median :-0.164  
 Mean   :0.501           Mean   :0.502   Mean   :-0.004  
 3rd Qu.:1.000           3rd Qu.:1.000   3rd Qu.: 0.885  
 Max.   :1.000           Max.   :1.000   Max.   : 1.584  
```

```
Sedentary_Hours_Per_Day     Income            BMI          Triglycerides
Min.   :-1.729          Min.   :-1.715   Min.   :-1.723   Min.   :-1.733
1st Qu.:-0.889          1st Qu.:-0.865   1st Qu.:-0.878   1st Qu.:-0.879
Median :-0.034          Median :-0.006   Median :-0.026   Median : 0.010
Mean   :-0.010          Mean   : 0.002   Mean   :-0.008   Mean   : 0.000
3rd Qu.: 0.865          3rd Qu.: 0.852   3rd Qu.: 0.853   3rd Qu.: 0.877
Max.   : 1.733          Max.   : 1.758   Max.   : 1.757   Max.   : 1.709
Physical_Activity_Days_Per_Week Sleep_Hours_Per_Day CountryArgentina
Min.   :-1.529                  Min.   :-1.521      Min.   :0.000
1st Qu.:-1.091                  1st Qu.:-1.018      1st Qu.:0.000
Median :-0.215                  Median :-0.012      Median :0.000
Mean   : 0.001                  Mean   :-0.005      Mean   :0.055
3rd Qu.: 1.100                  3rd Qu.: 0.994      3rd Qu.:0.000
Max.   : 1.538                  Max.   : 1.497      Max.   :1.000
CountryAustralia CountryBrazil   CountryCanada    CountryChina
Min.   :0.000    Min.   :0.000   Min.   :0.000   Min.   :0.00
1st Qu.:0.000    1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.00
Median :0.000    Median :0.000   Median :0.000   Median :0.00
Mean   :0.052    Mean   :0.053   Mean   :0.049   Mean   :0.05
3rd Qu.:0.000    3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:0.00
Max.   :1.000    Max.   :1.000   Max.   :1.000   Max.   :1.00
CountryColombia CountryFrance   CountryGermany   CountryIndia
Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000
1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000
Median :0.000   Median :0.000   Median :0.000   Median :0.000
Mean   :0.049   Mean   :0.052   Mean   :0.054   Mean   :0.043
3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:0.000
Max.   :1.000   Max.   :1.000   Max.   :1.000   Max.   :1.000
 CountryItaly    CountryJapan   CountryNew_Zealand CountryNigeria
Min.   :0.000   Min.   :0.000   Min.   :0.000      Min.   :0.000
1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000      1st Qu.:0.000
Median :0.000   Median :0.000   Median :0.000      Median :0.000
Mean   :0.049   Mean   :0.049   Mean   :0.051      Mean   :0.051
3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:0.000      3rd Qu.:0.000
Max.   :1.000   Max.   :1.000   Max.   :1.000      Max.   :1.000
CountrySouth_Africa CountrySouth_Korea  CountrySpain   CountryThailand
Min.   :0.000       Min.   :0.000      Min.   :0.000   Min.   :0.000
1st Qu.:0.000       1st Qu.:0.000      1st Qu.:0.000   1st Qu.:0.000
Median :0.000       Median :0.000      Median :0.000   Median :0.000
Mean   :0.049       Mean   :0.046      Mean   :0.049   Mean   :0.049
3rd Qu.:0.000       3rd Qu.:0.000      3rd Qu.:0.000   3rd Qu.:0.000
Max.   :1.000       Max.   :1.000      Max.   :1.000   Max.   :1.000
```

```
 CountryUnited_Kingdom CountryUnited_States CountryVietnam  ContinentAfrica
 Min.   :0.000         Min.   :0.000        Min.   :0.000   Min.   :0.0
 1st Qu.:0.000         1st Qu.:0.000        1st Qu.:0.000   1st Qu.:0.0
 Median :0.000         Median :0.000        Median :0.000   Median :0.0
 Mean   :0.053         Mean   :0.048        Mean   :0.049   Mean   :0.1
 3rd Qu.:0.000         3rd Qu.:0.000        3rd Qu.:0.000   3rd Qu.:0.0
 Max.   :1.000         Max.   :1.000        Max.   :1.000   Max.   :1.0
 ContinentAsia   ContinentAustralia ContinentEurope ContinentNorth_America
 Min.   :0.000   Min.   :0.000      Min.   :0.000   Min.   :0.000
 1st Qu.:0.000   1st Qu.:0.000      1st Qu.:0.000   1st Qu.:0.000
 Median :0.000   Median :0.000      Median :0.000   Median :0.000
 Mean   :0.287   Mean   :0.102      Mean   :0.257   Mean   :0.097
 3rd Qu.:1.000   3rd Qu.:0.000      3rd Qu.:1.000   3rd Qu.:0.000
 Max.   :1.000   Max.   :1.000      Max.   :1.000   Max.   :1.000
 ContinentSouth_America HemisphereNorthern_Hemisphere
 Min.   :0.000          Min.   :0.000
 1st Qu.:0.000          1st Qu.:0.000
 Median :0.000          Median :1.000
 Mean   :0.157          Mean   :0.643
 3rd Qu.:0.000          3rd Qu.:1.000
 Max.   :1.000          Max.   :1.000
 HemisphereSouthern_Hemisphere Heart_Attack_Risk
 Min.   :0.000                 Min.   :0.000
 1st Qu.:0.000                 1st Qu.:0.000
 Median :0.000                 Median :0.000
 Mean   :0.357                 Mean   :0.359
 3rd Qu.:1.000                 3rd Qu.:1.000
 Max.   :1.000                 Max.   :1.000
```

Cost Function

We are using a cost metric for classification of heart attack risk as 0 (no heart attack) or 1 (heart attack). In this case a false positive represents a prediction of a heart attack when there is no heart attack. A false negative represents a prediction of no heart attack when there is a heart attack. This model is meant to be a tool to see if you are at risk and NOT a diagnostic tool (we suggest this model be used for someone to assess whether they should get checked/monitored by a health care professional). Accordingly there is a low cost of a false positive related to a false negative since the only cost of a false positive is going to an appointment you don’t need to, but the cost of a false negative would be never seeking the help that may be needed.

```
#evaluate based on cost for the iterations
cost_FP <- 1 #predicting a heart attack when there is no heart attack
cost_FN <- 3 #predicting no heart attack when there is a heart attack
prob_thresh <- cost_FP/(cost_FN+cost_FP)

print(paste("The cost of a false positive is", cost_FP))
```

```
[1] "The cost of a false positive is 1"
```

```
print(paste("The cost of a false negative is", cost_FN))
```

```
[1] "The cost of a false negative is 3"
```

```
print(paste("This means a false negative is", cost_FN/cost_FP, "times worse than than
```

```
[1] "This means a false negative is 3 times worse than than a false positive"
```

```
print(paste("This sets the cost probability threshold as", prob_thresh))
```

```
[1] "This sets the cost probability threshold as 0.25"
```

Create the factor

This is a code detail to handle the binary output variable of the Heart_Attack_Risk variable

```
training_factor <- factor(train_output)
holdout_factor <- factor(holdout_outcome)
```

Model Evaluation Metric

The models we will generate will give probabilities as outputs. These then need to be thresholded by the cost probability threshold determined above to get a binary output of 0 or 1 corresponding to no risk or risk of heart attack. After the classification the predictions were compared to known data to calculate accuracy.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

For comparison of models a higher accuracy meant a better model performance.

Random Forest

A random forest model was fit. Tuning parameters were investigated to determine the optimal model in terms of accuracy.

Set the Tuning Parameter Ranges

The range over which to search to find the optimal values for mtry and minimum bucket size were set as a range of potential values.

```
#mtry range this should be close to sqrt(p) where p is 53.
mtry_range <- seq(4,50, by = 2)
#minimum bucket size searching from 1-10
min_bucket_range <- seq(1,4)
#set the number of trees to look at
num_trees = 1000
#number of times to run the random forest for each tuning set to determine the optimal
num_tuning = 5
```

The range over which to search to find the optimal values for mtry and minimum bucket size were set as a range of potential values. In the approach we set the number of trees to look at as a large value (1000) since this is where random forest tends to perform best. In addition we ran each random forest model 5 times to determine an averaged fit over multiple iterations.

Optimize the tuning parameters

An output variable is made which holds the mtry value, minimum bucket size value, and accuracy value for that model. For each value in the set range of mtry and min_bucket a random forest model is fit 5 separate times to the training data. The probabilities are output from the model and then split on the cost probability threshold set earlier. The accuracy is calculated with respect to the known heart attack risk values by creating a confusion matrix with the training data. The accuracy for each of the 5 iterations is averaged and the tuning parameters are stored along with the accuracy value.

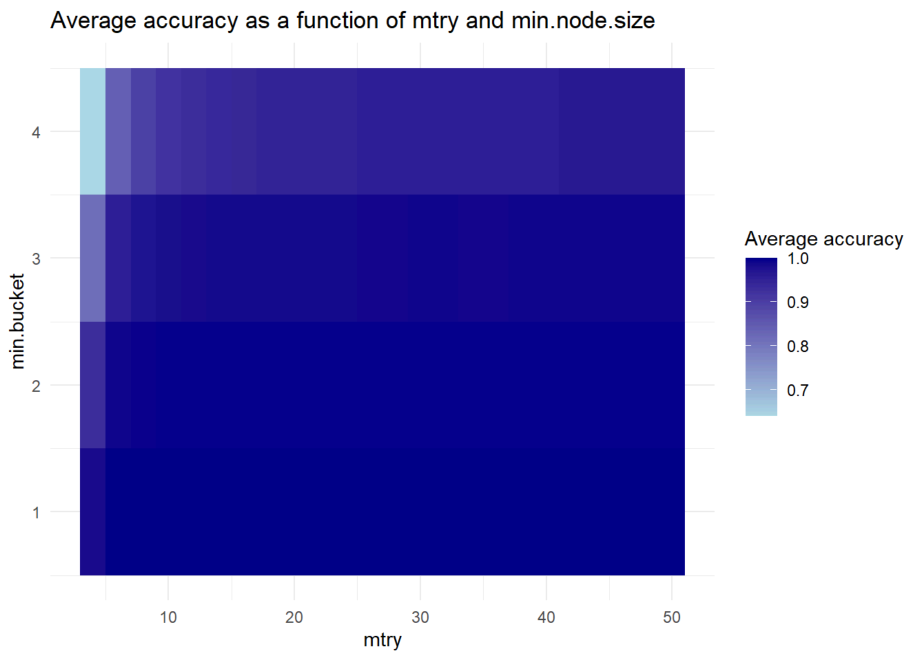
```
#initialize the output
output <- data.frame(mtry = numeric(length(mtry_range) * length(min_bucket_range)),
                     min_bucket = numeric(length(mtry_range) * length(min_bucket_range
                     accuracy = numeric(length(mtry_range) * length(min_bucket_range))
index <- 1

for (i in 1:length(mtry_range)) {
  #set the mtry here
  mtry_here <- mtry_range[i]
  for (j in 1:length(min_bucket_range)) {
    #set the min bucket size
    min_bucket_here <- min_bucket_range[j]
    #initialize each time something to hold the oob_mse value
    rf_accuracy_store <- numeric(num_tuning)

    #fit the random forest for this
    for (k in 1:num_tuning) {
      #create the model
      model <- ranger(Heart_Attack_Risk ~ ., data = training_data, num.trees = num_tre
      #make predictions on the probability
      predicted_prob <- predict(model, as.matrix(training_data))$predictions
      #turn the probability into a 1 or 0
      predicted_class <- ifelse(predicted_prob > prob_thresh, 1, 0)
      #calculate the accuracy
      predicted_factor <- factor(predicted_class, levels = levels(training_factor))
      rf_accuracy <- confusionMatrix(predicted_factor, training_factor)$overall["Accur
      rf_accuracy_store[k] <- rf_accuracy
    }
    #get the average of the rf accuracy
    avg_rf_accuracy <- mean(rf_accuracy_store)
    #print into the output dataframe
    output$mtry[index] = mtry_here
    output$min_bucket[index] = min_bucket_here
    output$accuracy[index] = avg_rf_accuracy
    #advance the index by 1
    index = index + 1
```

```
        }
      }
```

Plot the optimal tuning parameters

The tuning parameters are plotted here with the color representing the accuracy value. From this we can tell the optimal tuning parameters.

```
# Plot average accuracy as a function of mtry and min.node.size
ggplot(output, aes(x = mtry, y = min_bucket, fill = accuracy)) +
  geom_tile() +
  scale_fill_gradient(low = "lightblue", high = "darkblue", na.value = "grey50") +
  labs(title = "Average accuracy as a function of mtry and min.node.size",
       x = "mtry",
       y = "min.bucket",
       fill = "Average accuracy") +
  theme_minimal()
```



Identifying the best parameters

```
best_params <- output[which.max(output$accuracy), ]
print(paste("The optimal Mtry is:", best_params$mtry))
```

```
[1] "The optimal Mtry is: 46"
```

```
print(paste("The optimal minimum number of buckets is:", best_params$min_bucket))
```

```
[1] "The optimal minimum number of buckets is: 1"
```

Fit the best model

The random forest model is then fit once, with 1000 trees, on the training data with the optimized tuning parameters.

```
#train the random forest model
model_RF <- ranger(Heart_Attack_Risk ~ ., data = training_data, num.trees = num_trees,
```

The model is used to generate probabilities from the random forest model and the characteristics of these are summarized here.

```
#make predictions for the probability on the training set (leaving the holdout until l
probability_RF <- predict(model_RF, data = train_predictors)$predictions
print("The probabilites for the random forest model are summarized by:")
```

```
[1] "The probabilites for the random forest model are summarized by:"
```

```
print(summary(probability_RF))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.068   0.145   0.175   0.363   0.722   0.823 
```

The optimal accuracy from the training data fit was determined

```
print(paste("The optimal accuracy is:", best_params$accuracy))
```

```
[1] "The optimal accuracy is: 1"
```

Penalized Logistic Regression with Cross Validation

The next model that we chose to fit for our data was a penalized logistic regression model. We chose logistic regression since it is good for modeling a binary outcome.

The general process here is to:
1) tune alpha and lambda using training data.
2) Fit a model using glmnet with family = binomial.
3) Predict the training outcome using the training data as input. The accuracy of the LR model vs the training outcome is then calculated.
4) The LR model is stored for use in the ensemble model

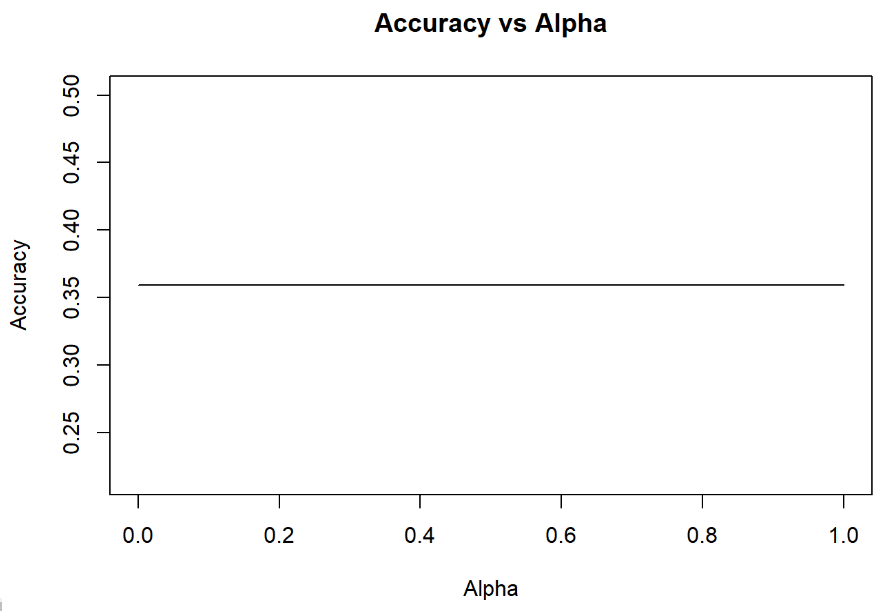
Tune the alpha

Alpha is a parameter that governs the type of regularization applied in the model fitting process. A value of 1 emphasizes sparsity (some coefficients become exactly zero), while a value of 0 emphasizes stability (all coefficients are shrunk towards zero, but none become exactly zero).

```
#-- Set folds so consistent over all runs
set.seed(4040)   # set seed to reproducible
K = 10           # number of folds
folds = rep(1:K, length=nrow(training_data)) %>% sample() # make folds

#-- Set alpha sequence
alpha_seq = seq(0, 1, by=.05)

lm_accuracy_store <- list()

for (iter in 1:length(alpha_seq)) {
  #get the alpha
  alpha_here <- alpha_seq[iter]
  #create the model
  model <- cv.glmnet(as.matrix(train_predictors), as.matrix(train_output),
          foldid = folds, alpha=alpha_here, family = binomial)
  #make predictions for the probability
  probability <- predict(model, as.matrix(train_predictors),s=0,type = "response")

  #Threshold the probability to a 1 or 0
  prediction <- ifelse(probability > prob_thresh, 1, 0)
  # Create prediction as factor for accuracy comparison
```

```
    prediction_factor <- factor(prediction, levels = levels(training_factor))
    # Calculate model accuracy for the given alpha
    lm_accuracy <- confusionMatrix(prediction_factor, training_factor)$overall["Accuracy
    # Store accuracy
    lm_accuracy_store[iter] <- lm_accuracy
  }
```

```
  # Plot alpha_seq vs lm_accuracy_store
  plot(alpha_seq, lm_accuracy_store, type = "l", xlab = "Alpha", ylab = "Accuracy", main
```



```
  optimal_alpha = alpha_seq[which.max(lm_accuracy_store)]
  print(paste("The optimal alpha value is:",optimal_alpha))
```

```
[1] "The optimal alpha value is: 0"
```

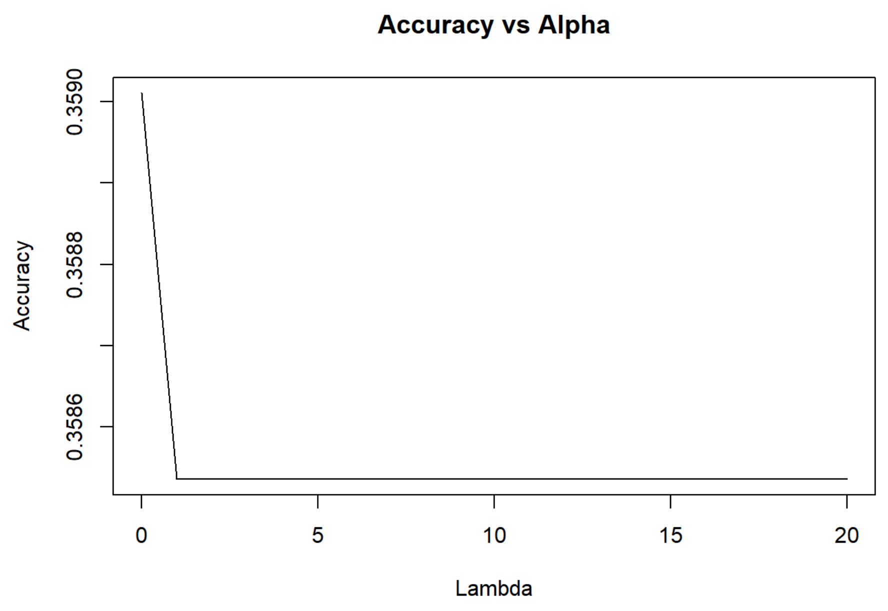
So from this we see that it is independent of alpha. So we choose an alpha value of 0, which retains all features but shrinks their weights toward 0.

Tune Lambda

A large lambda means high complexity/large weights for coefficients, depending on Lasso or Ridge regression. A lambda of 0 means that there is no restriction on the number of coefficients or weight sizes.

```
alpha = optimal_alpha
set.seed(4040)   # set seed to reproducible
K = 10           # number of folds
folds = rep(1:K, length=nrow(training_data)) %>% sample() # make folds

#-- Set alpha sequence
lambda_seq = seq(0, 20, by=1)

lm_accuracy_store <- list()

for (iter in 1:length(lambda_seq)) {
  #get the lambda
  lambda_here <- lambda_seq[iter]
  #create the model
  model <- cv.glmnet(as.matrix(train_predictors), as.matrix(train_output),
          foldid = folds, alpha=alpha,family = binomial)
  #make predictions for the probability
  probability <- predict(model, as.matrix(train_predictors), s=lambda_here,type = "res

  #convert the probability to a 1 or 0
  prediction <- ifelse(probability > prob_thresh, 1, 0)
  # Create prediction as factor for accuracy comparison
  prediction_factor <- factor(prediction, levels = levels(training_factor))
  # Calculate model accuracy for the given alpha
  lm_accuracy <- confusionMatrix(prediction_factor, training_factor)$overall["Accuracy
  # Store accuracy
  lm_accuracy_store[iter] <- lm_accuracy
}
```

```
# Plot lambda_seq vs lm_accuracy_store
plot(lambda_seq, lm_accuracy_store, type = "l", xlab = "Lambda", ylab = "Accuracy", ma
```



```
optimal_lambda = lambda_seq[which.max(lm_accuracy_store)]
print(paste("The optimal lambda value is:",optimal_lambda))
```

```
[1] "The optimal lambda value is: 0"
```

With the tuned alpha and lambda, fit the model

```
# Set alpha and lambda
alpha = optimal_alpha
lambda = optimal_lambda
# Fit the model
LR_model = glmnet(as.matrix(train_predictors), as.matrix(train_output), alpha=alpha,fa
```

```
#find the probabilities
probability_LR = predict(LR_model, newx=as.matrix(train_predictors), s = lambda,type =
print("The probabilites for the logistic regression model are summarized by:")
```

```
[1] "The probabilites for the logistic regression model are summarized by:"
```

```
print(summary(probability_LR))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.244   0.330   0.358   0.359   0.386   0.511 
```

Final Logistic Regression Model Accuracy

```
  #convert the probability to a 1 or 0
  prediction_lr <- ifelse(probability_LR > prob_thresh, 1, 0)
  # Create prediction as factor for accuracy comparison
  prediction_factor <- factor(prediction_lr, levels = levels(training_factor))
  # Calculate model accuracy for the given alpha
  accuracy_lr <- confusionMatrix(prediction_factor, training_factor)$overall["Accuracy

print(paste("The optimal logistic regression model accuracy:",accuracy_lr))
```

```
[1] "The optimal logistic regression model accuracy: 0.359010936757014"
```

For our specific dataset, our model tends to prefer mostly intercept based models with large number of coefficients with minimal penalty to parameter weight. The logistic regression model seems to struggle predicting the data since it is so complex, where logistic regression assumes a linear relationship between the predictor variables and the log-odds of the outcome.

Boosted Trees

We fit gradient boosted tree models using xgboost. We generated 500 base models. Each model was a simple decision tree that was then boosted 10 times, sequentially improving upon the mistakes of the previous model. Predictions for each patient was then made for all 500 models. Then, we generated a meta-model of these 500 predictions for each patient and fit a logistic regression model for predicting probabilities of HAR for each patient.

```
# Define a list to store base models
base_models <- list()

# Train 500 base models (XGBoost)
num_models = 500
for (i in 1:num_models) {
  base_models[[i]] <- xgboost(data = as.matrix(train_predictors), label = train_output
}
```

Then probabilities were determined from each of the base models from the training data.

```
  # Make predictions using base models
  train_probs <- matrix(NA, nrow = nrow(training_data), ncol = num_models)  # Initialize
  for (i in 1:num_models) {
    train_probs[, i] <- predict(base_models[[i]], as.matrix(train_predictors))
  }
```

The 500 models were then combined and a logistic regression was used to create a meta-model for the gradient boosted trees.

```
  # Combine base model probability and predicitons
  train_probability <- as.data.frame(train_probs)

  # Combine base model predictions and the true outcomes
  train_probability <- cbind(train_probability, Heart_Attack_Risk = train_output)

  # Define meta-model (logistic Regression)
  meta_model_BT <- train(Heart_Attack_Risk ~ ., data = train_probability, method = "glm"
```

```
Warning in train.default(x, y, weights = w, ...): You are trying to do
regression and your outcome only has two possible values Are you trying to do
classification? If so, use a 2 level factor as your outcome column.
```

Using the meta model the probabilities for the holdout data were also created in this step.

```
  # Make predictions on the holdout data using base models
  holdout_probs <- matrix(NA, nrow = nrow(holdout_data), ncol = num_models)  # Initializ
  for (i in 1:num_models) {
    holdout_probs[, i] <- predict(base_models[[i]], as.matrix(holdout_predictors))
  }
  # Combine base model predictions for test data
  holdout_probability <- as.data.frame(holdout_probs)
```

The meta model was then used with the training data to predict the probability from the boosted trees.

```
# Make probabilities using meta-model
probability_BT <- predict(meta_model_BT, newdata = train_probability)
```

Probabilities

```
print("The probabilites for the boosted trees model are summarized by:")
```

```
[1] "The probabilites for the boosted trees model are summarized by:"
```

```
print(summary(probability_BT))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.001   0.133   0.295   0.359   0.524   1.000 
```

Accuracy

The accuracy of the boosted trees meta-model was calculated by making the threshold, creating a confusion matrix and calculating accuracy

```
#convert the probability to a 1 or 0
prediction_bt <- ifelse(probability_BT > prob_thresh, 1, 0)
# Create prediction as factor for accuracy comparison
prediction_factor <- factor(prediction_bt, levels = levels(training_factor))
# Calculate model accuracy for the given alpha
accuracy_bt <- confusionMatrix(prediction_factor, training_factor)$overall["Accuracy"]
print(paste("The optimal boosted trees model accuracy:",accuracy_bt))
```

```
[1] "The optimal boosted trees model accuracy: 0.683626565224283"
```

A small investigation into Feature Importance

```
# Plotting feature importance
xgb.importance(model = base_models[[100]])  # You can replace base_models[[1]] with an
```

```
                             Feature    Gain    Cover Frequency
 1:                              BMI 0.11653 1.87e-01   0.10825
 2:              Exercise_Hours_Per_Week 0.09683 6.77e-02   0.09021
 3:              Sedentary_Hours_Per_Day 0.09127 8.87e-02   0.08247
 4:                        Triglycerides 0.08085 7.74e-02   0.07732
 5:                               Income 0.07540 1.45e-01   0.07474
 6:                          Cholesterol 0.07434 5.63e-02   0.08247
 7:                           Heart_Rate 0.07196 4.77e-02   0.06443
 8:                                  Age 0.06568 3.15e-02   0.07990
 9:                 Blood_Pressure_Upper 0.05850 5.95e-02   0.05928
10:                 Blood_Pressure_Lower 0.05375 5.96e-02   0.05670
11:                  Sleep_Hours_Per_Day 0.03122 3.48e-02   0.03351
12: Physical_Activity_Days_Per_Week 0.02813 2.58e-02   0.02320
13:                         Stress_Level 0.02604 1.38e-02   0.03093
14:                      CountryColombia 0.01108 2.44e-03   0.01031
15:                             Diabetes 0.00980 3.35e-02   0.01031
16:                          DietHealthy 0.00919 1.25e-03   0.01031
17:                          DietAverage 0.00806 1.12e-03   0.00515
18:                  Alcohol_Consumption 0.00800 3.80e-03   0.00773
19:                         CountrySpain 0.00797 6.88e-04   0.00773
20:                      ContinentEurope 0.00788 1.43e-02   0.00773
21:                         CountryIndia 0.00552 8.86e-03   0.00515
22:               ContinentNorth_America 0.00541 6.35e-04   0.00515
23:                      CountryThailand 0.00540 1.03e-03   0.00773
24:                        CountryBrazil 0.00525 6.61e-04   0.00258
25:                         CountryItaly 0.00487 2.57e-04   0.00515
26:                   CountryNew_Zealand 0.00413 3.89e-04   0.00515
27:                CountryUnited_Kingdom 0.00400 2.25e-04   0.00515
28:                   CountrySouth_Korea 0.00374 1.66e-02   0.00515
29:                       CountryNigeria 0.00347 1.30e-02   0.00258
30:                        DietUnhealthy 0.00312 3.17e-05   0.00258
31:                              Obesity 0.00310 2.75e-04   0.00258
32:                      ContinentAfrica 0.00298 1.79e-03   0.00258
33:                 CountryUnited_States 0.00276 3.99e-04   0.00258
34:              Previous_Heart_Problems 0.00255 1.80e-03   0.00258
35:                  CountrySouth_Africa 0.00216 1.63e-03   0.00515
36:                         CountryChina 0.00206 2.20e-04   0.00258
37:               ContinentSouth_America 0.00202 4.23e-05   0.00258
38:                   ContinentAustralia 0.00173 7.41e-05   0.00258
39:                       CountryGermany 0.00151 1.03e-04   0.00258
40:                              Smoking 0.00104 3.44e-05   0.00258
```

```
41:                  Family_History 0.00071 1.59e-05   0.00258
                            Feature     Gain     Cover Frequency
```

```
summary(probability_BT)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.001   0.133   0.295   0.359   0.524   1.000 
```

Model Averaging

We are assembling our three models to create an averaged model using all of them. The final model form will be:

$$Probability_{overall} = A * Probability_{LR} + B * Probability_{RF} + C * Probability_{BT}$$

## Coefficient Combinations To determine the final model we sought to determine the values of A, B, and C that maximized the accuracy of the averaged model on the training data.A represents the contribution of the logistic regression, b the contribution of the random forest, and C the contribution of the boosted trees. A, B, and C were set such that:

$$A = [0 : 0.01 : 1] B = [0 : 0.01 : 1] C = [0 : 0.01 : 1]$$

AND

$$A + B + C = 1$$

```
A <- seq(0, 1, by = 0.01)
B <- seq(0, 1, by = 0.01)
C <- seq(0, 1, by = 0.01)

# Generate all combinations of A, B, and C
combinations <- expand.grid(A = A, B = B, C = C)

# Filter combinations where A + B + C equals 1
valid_combinations <- subset(combinations, A + B + C == 1)
print(paste("There are",nrow(valid_combinations),"combinations of A, B, and C that wer
```

```
[1] "There are 5027 combinations of A, B, and C that were tested"
```

Determining the optimal coefficients

We used each of these combinations of coefficients to create an averaged model. Using that averaged model we got averaged probabilities. These probabilities were thresholded as done elsewhere and then accuracy was calculated for each model.

```
#iterate over the valid combinations to decide the optimal values of A B C
output <- list()
for(i in 1:nrow(valid_combinations)) {
  A_here = valid_combinations$A[i]
  B_here = valid_combinations$B[i]
  C_here = valid_combinations$C[i]
  #get the averaged model probabilities
  averaged_model_probability = A_here*probability_LR + B_here*probability_RF + C_here*
  #get the averaged model predictions
  averaged_model_predictions <- ifelse(averaged_model_probability > prob_thresh, 1, 0)
  #get the accuracy of the data
  predicted_factor <- factor(averaged_model_predictions, levels = levels(training_fact

  accuracy <- confusionMatrix(predicted_factor, training_factor)$overall["Accuracy"]
  #store the data
  output[i] <- accuracy
}
```

```
best_index <- which.max(output)
A_best <- valid_combinations$A[best_index]
B_best <- valid_combinations$B[best_index]
C_best <- valid_combinations$C[best_index]
print(paste("There best A value is", A_best))
```

```
[1] "There best A value is 0.05"
```

```
print(paste("There best B value is", B_best))
```

```
[1] "There best B value is 0.95"
```

```
print(paste("There best C value is", C_best))
```

```
[1] "There best C value is 0"
```

A large value here represents a large contribution of the model. Since the optimal value of B is 0.95, the random forest informs most of this model. On the opposite, since the optimal value of C is 0 the boosted trees model is not used in the model at all.

This approach provides an excellent automatic process for combining the models into one ensemble model, but it has one drawback. It uses the training data (which all the models were tuned for) for calculating the accuracy for each model. Thus, this process will give a high priority to any model that is hyper-tuned to the training data. Improving this process could help our predictions for the holdout data. One solution that could have helped here would be to take a second set of holdout data that could be used to score the models' performance and generate the ensemble model weights.

Accuracy of Averaged Model

The accuracy score was determined for the averaged model with the tuned coefficients on the training data.

```
accuracy_overall = output[[which.max(output)]]
print(paste("The optimal accuracy is:", accuracy_overall))
```

```
[1] "The optimal accuracy is: 0.999841496275163"
```

This accuracy tells us that this averaged model is able to represent the data very well.

Evaluate All Models on the Holdout Data

Generate holdout data probabilities

For each model the holdout probabilities were generated. The outcomes (accuracy and confusion matrix) between each model are compared at the end of this section.

```
#need to generate the probabilities for LR on the holdout data
holdout_probability_LR = predict(LR_model, newx=as.matrix(holdout_predictors), s = laml

#need to generate the probabilities for RF on the holdout data
holdout_probability_RF <- predict(model_RF, data = holdout_predictors)$predictions
```

```
#need to generate the probabilities for BT on the holdout data
holdout_probability_BT <- predict(meta_model_BT, newdata = holdout_probability)
```

Evaluate the logistic regression model on the holdout data

The same process as before was used to generate the predictions and accuracy of the logistic regression model on the holdout data.

```
#threshold the probabilities
holdout_LR_predictions <- ifelse(holdout_probability_LR > prob_thresh, 1, 0)
#now evaluate the accuracy of this model
predicted_factor_LR <- factor(holdout_LR_predictions, levels = levels(holdout_factor))
#Get the confusion matrix
conf_matrix_LR <- confusionMatrix(predicted_factor_LR, holdout_factor)
#Calculate accuracy
holdout_LR_accuracy <- conf_matrix_LR$overall["Accuracy"]
```

The confusion matrix for the Logistic regression model is:

```
print(conf_matrix_LR$table)
```

```
          Reference
Prediction   0   1
         0   1   0
         1 458 242
```

```
print(paste("The accuracy on the holdout data for the logistic regression model is", h
```

```
[1] "The accuracy on the holdout data for the logistic regression model is
0.346647646219686"
```

Evaluate the random forest model on the holdout data

The same process as before was used to generate the predictions and accuracy of the random forest model on the holdout data.

```r
#threshold the probabilities
holdout_RF_predictions <- ifelse(holdout_probability_RF > prob_thresh, 1, 0)
#now evaluate the accuracy of this model
predicted_factor_RF <- factor(holdout_RF_predictions, levels = levels(holdout_factor))
#Get the confusion matrix
conf_matrix_RF <- confusionMatrix(predicted_factor_RF, holdout_factor)
#Calculate accuracy
holdout_RF_accuracy <- conf_matrix_RF$overall["Accuracy"]
```

The confusion matrix for the random forest model is:

```r
print(conf_matrix_RF$table)
```

```
          Reference
Prediction   0   1
         0   9   4
         1 450 238
```

```r
print(paste("The accuracy on the holdout data for the random forest model is", holdout_
```

```
[1] "The accuracy on the holdout data for the random forest model is 0.352353780313837"
```

Evaluate the boosted trees model on the holdout data

The same process as before was used to generate the predicitons and accuracy of the boosted trees meta-model on the holdout data.

```r
#threshold the probabilities
holdout_BT_predictions <- ifelse(holdout_probability_BT > prob_thresh, 1, 0)
#now evaluate the accuracy of this model
predicted_factor_BT <- factor(holdout_BT_predictions, levels = levels(holdout_factor))
#Get the confusion matrix
conf_matrix_BT <- confusionMatrix(predicted_factor_BT, holdout_factor)
#Calculate accuracy
holdout_BT_accuracy <- conf_matrix_BT$overall["Accuracy"]
```

The confusion matrix for the random forest model is:

```
print(conf_matrix_BT$table)
```

```
          Reference
Prediction   0   1
         0 191 102
         1 268 140
```

Comparing the confusion matrices and accuracy scores across the models: * The random forest model tended to predict very few negative cases. It was certainly overfit to the training data, and scored poorly on the holdout compared to its great performance on the training data. * The logistic regression model was consistent between the holdout and training data predictions. It made very few negative case predictions (similar to RF), and interestingly had similar performance to the random forest model. * The Boosted tree model had the highest accuracy on the holdout data. This model was far more likely to predict a negative case than the other models, with many of those times it was correct. It did however have quite a few false negatives, which is something we were hoping to avoid.

```
print(paste("The accuracy on the holdout data for the boosted trees model is", holdout_
```

```
[1] "The accuracy on the holdout data for the boosted trees model is 0.472182596291013"
```

Assemble the Final Averaged Model

The final averaged model with the optimized coefficients was assembled. Probabilities and predictions were generated on the holdout data.

```
#final model probability
final_model_probability <- A_best*holdout_probability_LR + B_best*holdout_probability_
#final model predictions
final_model_predictions <- ifelse(final_model_probability > prob_thresh, 1, 0)

#now evaluate the accuracy of this model
predicted_factor <- factor(final_model_predictions, levels = levels(holdout_factor))

#get the accuracy of this one
conf_matrix <- confusionMatrix(predicted_factor, holdout_factor)
```

The confuction matrix

```
print(conf_matrix$table)
```

```
          Reference
Prediction   0   1
         0   7   4
         1 452 238
```

```
accuracy <- conf_matrix$overall["Accuracy"]
print(paste("The overall model accuracy is", accuracy))
```

```
[1] "The overall model accuracy is 0.349500713266762"
```

When our ensemble model was scored, we got an accuracy score of 0.35, which is not ideal. Looking at the confusion matrix though we see that most of the errors are in the false positive, which signifies our model is cautious, which is not a bad thing. The result we wanted to avoid was false negatives, and here there were very few.

Summary and Conclusions

From this process we gained insights into the strengths and weaknesses of the models:
* The Random Forest model preferred deep, complex trees. This makes this model extremely good at predicting the training data, but it overfit for the holdout data. This make sense, since our process tuned for the highest possible accuracy for the RF model. If we were to process this data again, we should not tune so aggressively. * The Logistic Regression model preferred large intercepts and a greater number of parameters. We believe our data was too complex for LR to do well making predictions since LR assumes a linear relationship between the predictor variables and the log-odds of the outcome. It was a consistent predictor in the threshold probability region, so it seemsl like it probably added some stability to the predictions and was given a small weight. * The Boosted Tree model had the best holdout performance but was given less weight than RF for ensemble model, likely because it wasn't as finely tuned to the training data. We hypothesize that the boosted tree model was generating similar predictions to the RF model, but less aggressively. So the RF model was given all the weight. The boosted tree model gave a balanced approach for our data overall.

There are several lessons learned from this project: * Data engineering was crucial for our dataset, and each improvement we made here saw improvements in model performance. We believe there are probably further insights that could be gained by further refining and preprocessing the data. * We learned that we had the pay careful attention to whether our process/function was outputting predictions or probabilities. It was easy to accidentally compare predictions to probabilities or vice versa.