Predicting Heart Disease with R and Caret

Eduardo Abdala

Introduction

Diagnosing disease and being able to take preventative measures is no easy task. When we talk about heart disease, we are usually told to eat healthier, exercise, and minimize stress. However, every person is different and a more targeted approach can improve patient outcomes. Clinicians and healthcare managers can leverage machine learning to improve patient care in all aspects of the service life-cycle.

Problem

Diagnosing heart disease is not as easy as doing blood work and looking at an EKG. We are given a data set with de-identified patient records (detailed below) and we are tasked with creating a machine learning model that predicts if a patient has heart disease.

Project Goals

Predict which patients are more likely to have heart disease, given a number of possible predictors. This tool would be used to support and streamline the decision making process for medical professionals. The following steps were taken:

1. Provide exploratory data analysis
2. Provide an overview of the data with visualizations
3. Preprocess the data set (partition, encode categorical features, and normalize)
4. Train and test different classification machine learning algorithms
5. Provide performance metrics and interpret the results.

Description of the Data

This data set is taken from kaggle.com and sourced from the UC Irvine Machine Learning Repository. We have a data set of n = 303 with 13 features and one target column used to train a classifier machine learning algorithm.

The features are as follows:

1. age
2. sex (male = 1, female = 0)
3. cp = chest pain type (values 0-3)
4. trestbps = resting blood pressure (mm Hg)
5. chol = serum cholesterol (mg/dl)
6. fbs = fasting blood sugar (> 120 mg/dl)
7. restecg = resting ECG results (values 0-2)
8. thalach = max heart rate achieved
9. exang = exercised induced angina
10. oldpeak = ST depression induced by exercise

11. slope = slope of the peak exercise ST segment (values 0-2)
12. ca = number of major vessels colored by flouroscopy (values 0-4)
13. thal: 1 = normal, 2 = fixed defect, 3 = reversible defect
14. target = presence of heart disease (absent = FALSE, present = TRUE)

Analysis was done in RStudio using R v4.0.3. Package versions were managed with renv.

```
dplyr::glimpse(heart)
```

```
## Rows: 303
## Columns: 14
## $ age      <dbl> 63, 37, 41, 56, 57, 57, 56, 44, 52, 57, 54, 48, 49, 64, 58, 5~
## $ sex      <fct> male, male, female, male, female, male, female, male, male, m~
## $ cp       <fct> 3, 2, 1, 1, 0, 0, 1, 1, 2, 2, 0, 2, 1, 3, 3, 2, 2, 3, 0, 3, 0~
## $ trestbps <dbl> 145, 130, 130, 120, 120, 140, 140, 120, 172, 150, 140, 130, 1~
## $ chol     <dbl> 233, 250, 204, 236, 354, 192, 294, 263, 199, 168, 239, 275, 2~
## $ fbs      <fct> 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0~
## $ restecg  <fct> 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1~
## $ thalach  <dbl> 150, 187, 172, 178, 163, 148, 153, 173, 162, 174, 160, 139, 1~
## $ exang    <fct> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0~
## $ oldpeak  <dbl> 2.3, 3.5, 1.4, 0.8, 0.6, 0.4, 1.3, 0.0, 0.5, 1.6, 1.2, 0.2, 0~
## $ slope    <fct> 0, 0, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 0, 2, 2, 1~
## $ ca       <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0~
## $ thal     <fct> 1, 2, 2, 2, 2, 1, 2, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3~
## $ target   <fct> positive, positive, positive, positive, positive, positive, p~
```

Below is an overview of our data set. This view is helpful for us to see feature distributions, class imbalances in our target column (which can influence what we do in preprocessing and algorithm selection), and get a feel for variable distributions. I renamed the target and sex columns to make them easier to interpret. Instead of 0 and 1, words are used (negative, positive, male, female).

```
summary(heart)
```

```
##       age            sex      cp         trestbps          chol       fbs
##  Min.   :29.00   female: 96   0:143   Min.   : 94.0   Min.   :126.0   1: 45
##  1st Qu.:47.50   male  :207   1: 50   1st Qu.:120.0   1st Qu.:211.0   0:258
##  Median :55.00                2: 87   Median :130.0   Median :240.0
##  Mean   :54.37                3: 23   Mean   :131.6   Mean   :246.3
##  3rd Qu.:61.00                        3rd Qu.:140.0   3rd Qu.:274.5
##  Max.   :77.00                        Max.   :200.0   Max.   :564.0
##  restecg    thalach      exang      oldpeak     slope   ca      thal
##  0:147   Min.   : 71.0   0:204   Min.   :0.00   0: 21   0:175   0:  2
##  1:152   1st Qu.:133.5   1: 99   1st Qu.:0.00   1:140   2: 38   1: 18
##  2:  4   Median :153.0           Median :0.80   2:142   1: 65   2:166
##          Mean   :149.6           Mean   :1.04           3: 20   3:117
##          3rd Qu.:166.0           3rd Qu.:1.60           4:  5
##          Max.   :202.0           Max.   :6.20
##       target
##  negative:138
##  positive:165
##
##
##
##
```

Grouping our patients by sex, we find that our sample set has a large number of women with heart disease, compared to men. This is an interesting find that could be used in our predictions. An important characteristic

of a dataset to analyze when solving classification problems is class imbalance. Moreover, the variable that you are trying to pick should be as balanced as possible between the different possible outcomes. We calculate the percentage of patients with heart disease to be 54.46% and those without to be 45.54%. This is not considered imbalanced. In other datasets, one class can be significantly over-represented in the sample. making it more difficult for your algorithm to learn the characteristics of the underrepresented class. This can be toned down by upsampling or downsampling, though I don't suspect that will yield significant model improvements here.

```
# clean with dplyr
heart %>%
  count(target, sex)
```

```
## # A tibble: 4 x 3
##   target   sex        n
##   <fct>    <fct>  <int>
## 1 negative female    24
## 2 negative male     114
## 3 positive female    72
## 4 positive male      93
```

```
targetsex_counts <- heart %>%
                      count(target, sex)

print(c("Percent of people with heart disease:",paste(round(target_counts$n[2]/nrow(heart)*100, 2), "%"
```

```
## [1] Percent of people with heart disease: 54.46%
```

```
print(c("Percent of people without heart disease:",paste(round(target_counts$n[1]/nrow(heart)*100, 2),
```

```
## [1] Percent of people without heart disease:
## [2] 45.54%
```

```
## [1] Percent of men with heart disease: 44.93%
```

```
## [1] Percent of women with heart disease: 75%
```

```
## [1] Overall percent male in sample: 68.32%
```

```
## [1] Overall percent female in sample: 31.68%
```

Here, we examine the difference in mean values for the continuous variables to see if any stand out (chol, thalach, and oldpeak). We also see that people with heart disease were usually a few years younger. We usually think that you are more susceptible to heart disease as you get older. While there are differences in means in this sample, we do not know if these are due to chance. Hypothesis testing would allow us to see if there is evidence to support a real difference between the two groups. I explored if these changes were significant enough in another paper in this repository.

```
meanby_target <- heart %>%
                  group_by(target) %>%
                  summarize_at(c("age", "trestbps", "chol", "thalach", "oldpeak"),mean)
meanby_target
```

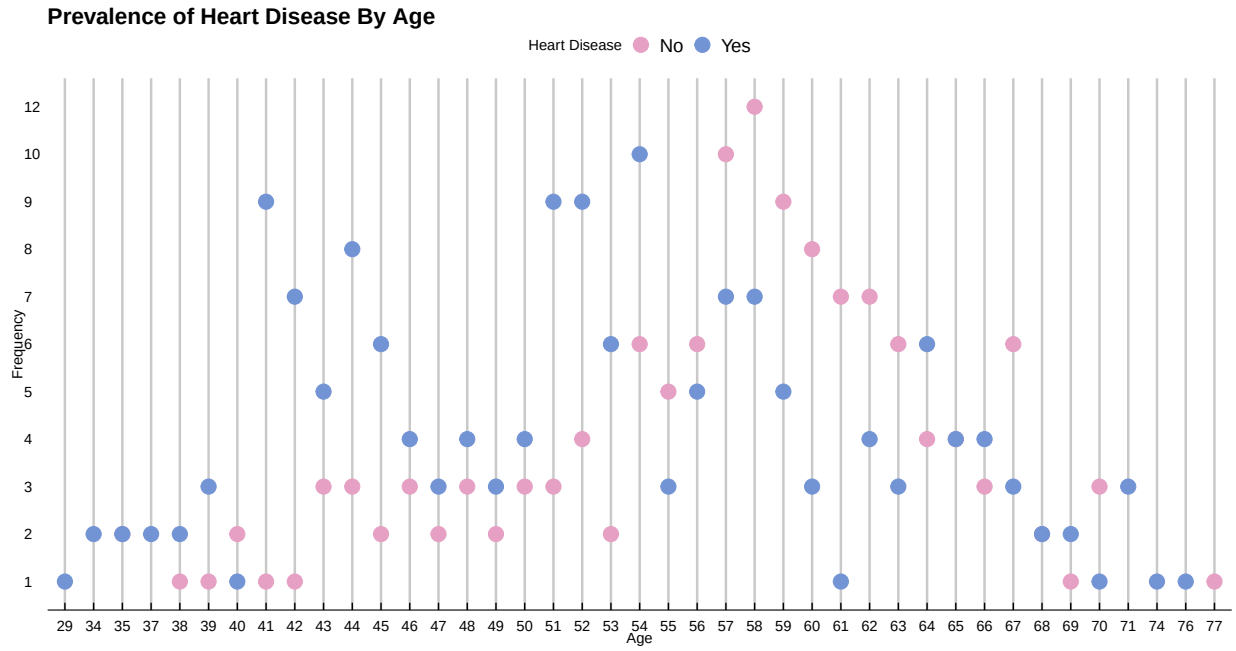
```
## # A tibble: 2 x 6
##   target     age trestbps  chol thalach oldpeak
##   <fct>    <dbl>    <dbl> <dbl>   <dbl>   <dbl>
## 1 negative  56.6     134.  251.    139.   1.59
## 2 positive  52.5     129.  242.    158.   0.583
```

We could even separate the data by sex to tell us if there are differences between males and females in the mean values above for our continuous variables.

```
meanby_targetsex <- heart %>%
  group_by(target,sex) %>%
  summarize_at(c("age", "trestbps", "chol", "thalach", "oldpeak"),mean) %>%
  arrange(sex)

meanby_targetsex
```

```
## # A tibble: 4 x 7
## # Groups:   target [2]
##   target   sex      age trestbps  chol thalach oldpeak
##   <fct>    <fct>  <dbl>    <dbl> <dbl>   <dbl>   <dbl>
## 1 negative female  59.0     146.  275.    142.   1.84
## 2 positive female  54.6     129.  257.    154.   0.554
## 3 negative male    56.1     132.  246.    138.   1.53
## 4 positive male    50.9     130.  231.    162.   0.605
```



The graph above visualizes the difference in age distributions between the heart disease group and the healthy group. We again see that a number of patients with heart disease were younger, which goes against what we normally think. We can hypothesize from this graph that age will be less useful in predicting heart disease, though we will be able to visualize variable importance below.

Preprocessing

We are ready to prepare our data for machine learning algorithms. We first separate our data into a set that we will use to train, or teach our computer what numbers a heart disease patient usually has, then another to test for its accuracy. We have to use a test set to make sure that the algorithm can interpret and predict using records it has never seen before. Then, we convert our qualitative variables into numbers using one-hot encoding so our computer can work with them Lastly, we normalize the data so variables like trestbps do not mathematically overpower variables like oldpeak due to the difference in magnitude.

```r
#partition, encode, normalize both train and test sets

set.seed(100)
trainIndex <- createDataPartition(heart$target, p = .70, list = FALSE)
trainingSet <- heart[trainIndex,]
testSet<- heart[-trainIndex,]

dummies <- dummyVars(target ~ ., data = trainingSet)
trainingSetX <- as.data.frame(predict(dummies, newdata = trainingSet))
```

```
## Warning in model.frame.default(Terms, newdata, na.action = na.action, xlev =
## object$lvls): variable 'target' is not a factor
```

```r
normModel <- preProcess(trainingSetX, method = "center")
trainingSetX <- predict(normModel, newdata = trainingSetX)
head(trainingSetX)
```

```
##           age sex.female  sex.male       cp.0       cp.1       cp.2       cp.3
## 1   9.1830986  -0.314554  0.314554 -0.4788732 -0.1737089 -0.2816901  0.9342723
## 2 -12.8169014   0.685446 -0.685446 -0.4788732  0.8262911 -0.2816901 -0.0657277
## 3   3.1830986   0.685446 -0.685446  0.5211268 -0.1737089 -0.2816901 -0.0657277
## 4   2.1830986   0.685446 -0.685446 -0.4788732  0.8262911 -0.2816901 -0.0657277
## 5  -9.8169014  -0.314554  0.314554 -0.4788732  0.8262911 -0.2816901 -0.0657277
## 6   0.1830986  -0.314554  0.314554  0.5211268 -0.1737089 -0.2816901 -0.0657277
##      trestbps       chol      fbs.1      fbs.0  restecg.0  restecg.1
## 1  14.1126761 -12.549296  0.8638498 -0.8638498  0.5399061 -0.5258216
## 2  -0.8873239 -41.549296 -0.1361502  0.1361502  0.5399061 -0.5258216
## 3 -10.8873239 108.450704 -0.1361502  0.1361502 -0.4600939  0.4741784
## 4   9.1126761  48.450704 -0.1361502  0.1361502  0.5399061 -0.5258216
## 5 -10.8873239  17.450704 -0.1361502  0.1361502 -0.4600939  0.4741784
## 6   9.1126761  -6.549296 -0.1361502  0.1361502 -0.4600939  0.4741784
##     restecg.2    thalach    exang.0    exang.1    oldpeak    slope.0    slope.1
## 1 -0.01408451 -0.5211268  0.3380282 -0.3380282  1.2441315  0.9342723 -0.4460094
## 2 -0.01408451 21.4788732  0.3380282 -0.3380282  0.3441315 -0.0657277 -0.4460094
## 3 -0.01408451 12.4788732 -0.6619718  0.6619718 -0.4558685 -0.0657277 -0.4460094
## 4 -0.01408451  2.4788732  0.3380282 -0.3380282  0.2441315 -0.0657277  0.5539906
## 5 -0.01408451 22.4788732  0.3380282 -0.3380282 -1.0558685 -0.0657277 -0.4460094
## 6 -0.01408451  9.4788732  0.3380282 -0.3380282  0.1441315 -0.0657277 -0.4460094
##      slope.2      ca.0       ca.2       ca.1        ca.3        ca.4
## 1 -0.4882629 0.3896714 -0.1126761 -0.1971831 -0.06103286 -0.01877934
## 2  0.5117371 0.3896714 -0.1126761 -0.1971831 -0.06103286 -0.01877934
## 3  0.5117371 0.3896714 -0.1126761 -0.1971831 -0.06103286 -0.01877934
## 4 -0.4882629 0.3896714 -0.1126761 -0.1971831 -0.06103286 -0.01877934
## 5  0.5117371 0.3896714 -0.1126761 -0.1971831 -0.06103286 -0.01877934
## 6  0.5117371 0.3896714 -0.1126761 -0.1971831 -0.06103286 -0.01877934
##         thal.0      thal.1     thal.2     thal.3
## 1 -0.004694836  0.93896714 -0.5258216 -0.4084507
## 2 -0.004694836 -0.06103286  0.4741784 -0.4084507
## 3 -0.004694836 -0.06103286  0.4741784 -0.4084507
## 4 -0.004694836 -0.06103286  0.4741784 -0.4084507
## 5 -0.004694836 -0.06103286 -0.5258216  0.5915493
## 6 -0.004694836 -0.06103286  0.4741784 -0.4084507
```

```r
trainingSet <- cbind(trainingSet$target, trainingSetX)
names(trainingSet)[1] <- "target"
```

```
testSet_dummy <- predict(dummies, testSet)
```

```
## Warning in model.frame.default(Terms, newdata, na.action = na.action, xlev =
## object$lvls): variable 'target' is not a factor
```

```
testSet_norm <- predict(normModel, testSet_dummy)
testSet_norm <- data.frame(testSet_norm)
testSet <- cbind(testSet$target, testSet_norm)
names(testSet) <- names(trainingSet)
```

Random Forest

Here, we create a random forest object and expose it to the training set.

```
set.seed(100)
rf <- caret::train(target ~., data = trainingSet, method = "rf")
rf
```

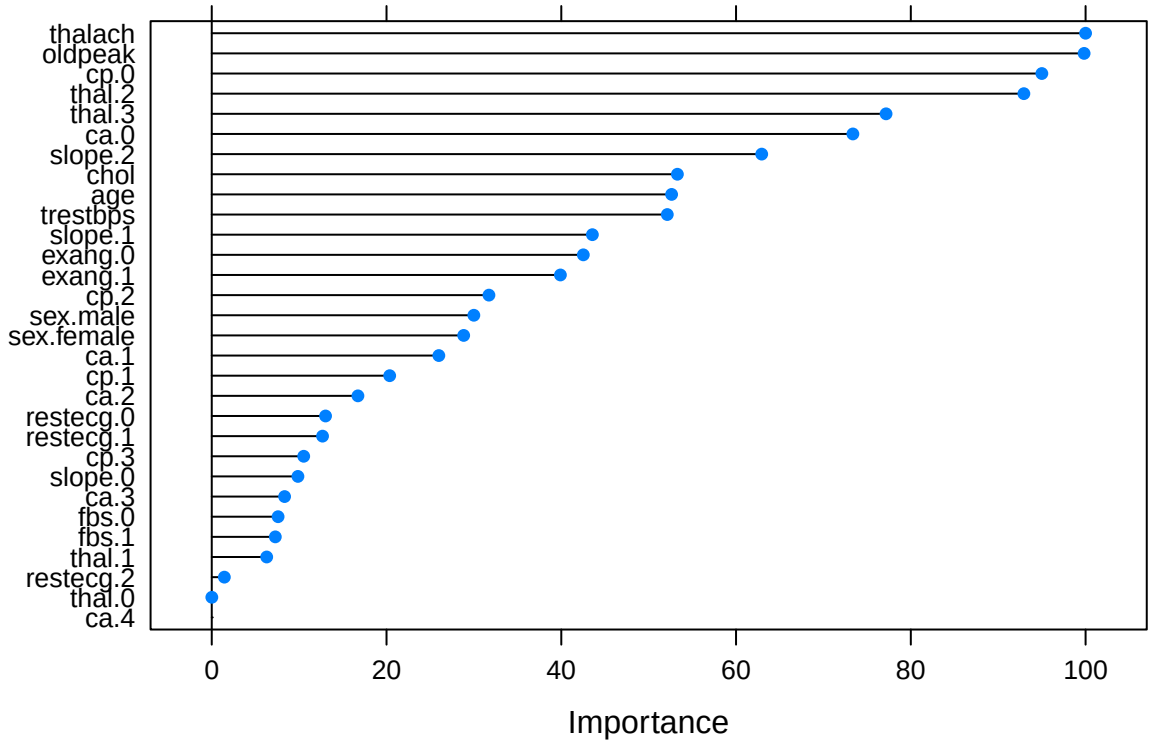
```
## Random Forest
##
## 213 samples
##  30 predictor
##   2 classes: 'negative', 'positive'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 213, 213, 213, 213, 213, 213, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8259955  0.6484179
##   16    0.7932183  0.5812017
##   30    0.7744927  0.5423991
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Our output shows a top accuracy of 82.6% without any parameter tuning (mtry). Below, we see what the model considered to be the most important factors when predicting heart disease. Highest on the list is are our thalach and oldpeak variables. The model is saying that these two variables affected the prediction output the most. A thing to note here is that this procedure likes to favor continuous variables more than discrete ones. Further collaboration with medical experts would test if this finding makes sense.

We then tested the algorithm's accuracy with the test set and got 85.56% accuracy. With the confusion matrix, we can see a breakdown of where the correct and incorrect predictions were.

Sensitivity is how well our model was able to identify the presence of heart disease in our patients. In its current state, the model correctly identified heart disease 92% of the time. Conversely, specificity is how often the model was able to know the patient did not have heart disease. Here, it was 78% of the time.

```
varimp_RF <- varImp(rf)
plot(varimp_RF, main = "Heart Disease Variable Importance (Random Forest)")
```




```
fitted <- predict(rf, testSet)
confusionMatrix(reference = testSet$target, data = fitted, mode = "everything", positive = "positive")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction negative positive
##   negative       32        4
##   positive        9       45
##
##                Accuracy : 0.8556
##                  95% CI : (0.7657, 0.9208)
##     No Information Rate : 0.5444
##     P-Value [Acc > NIR] : 3.463e-10
##
##                   Kappa : 0.7059
##
##  Mcnemar's Test P-Value : 0.2673
##
##             Sensitivity : 0.9184
##             Specificity : 0.7805
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.8889
##               Precision : 0.8333
##                  Recall : 0.9184
##                      F1 : 0.8738
```

```
##              Prevalence : 0.5444
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.8494
##
##        'Positive' Class : positive
##
```

We can see above that our accuracy with the test dataset (patient records the model has never seen) is 85.6%. This means that we did not run in to the problem of overfitting, where predictions on the training set are much better than on new data. Then we do some parameter tuning with a 5-fold cross validation, exposing our model to many different test sets to make sure we are getting a reasonable accuracy score.

```
twoClassCtrl <- trainControl(
  method = "repeatedcv",
  number = 5,
  repeats = 5,
  savePredictions = "final",
  classProbs = TRUE,
  summaryFunction = twoClassSummary
)
```

```
set.seed(100)
rfTL <- train(target ~., data = trainingSet, method = "rf", metric = "ROC", trControl = twoClassCtrl, tu
fittedTL <- predict(rfTL, testSet)
confusionMatrix(reference = testSet$target, data = fittedTL, mode = "everything", positive = "positive")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction negative positive
##   negative       30        4
##   positive       11       45
##
##                Accuracy : 0.8333
##                  95% CI : (0.74, 0.9036)
##     No Information Rate : 0.5444
##     P-Value [Acc > NIR] : 7.067e-09
##
##                   Kappa : 0.6593
##
##  Mcnemar's Test P-Value : 0.1213
##
##             Sensitivity : 0.9184
##             Specificity : 0.7317
##          Pos Pred Value : 0.8036
##          Neg Pred Value : 0.8824
##               Precision : 0.8036
##                  Recall : 0.9184
##                      F1 : 0.8571
##              Prevalence : 0.5444
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.6222
##       Balanced Accuracy : 0.8250
##
```

```
##        'Positive' Class : positive
##
```

```
rfGrid <- data.frame(mtry = c(2, 5, 7, 9, 10, 11, 12, 13, 15, 17, 19))

rfTG <- train(target ~., data = trainingSet, method = "rf", metric = "ROC", trControl = twoClassCtrl, tu

fittedTG <- predict(rfTG, testSet)
confusionMatrix(reference = testSet$target, data = fittedTG, mode = "everything", positive = "positive")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction negative positive
##   negative       31        4
##   positive       10       45
##
##                Accuracy : 0.8444
##                  95% CI : (0.7528, 0.9123)
##     No Information Rate : 0.5444
##     P-Value [Acc > NIR] : 1.629e-09
##
##                   Kappa : 0.6826
##
##  Mcnemar's Test P-Value : 0.1814
##
##             Sensitivity : 0.9184
##             Specificity : 0.7561
##          Pos Pred Value : 0.8182
##          Neg Pred Value : 0.8857
##               Precision : 0.8182
##                  Recall : 0.9184
##                      F1 : 0.8654
##              Prevalence : 0.5444
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.6111
##       Balanced Accuracy : 0.8372
##
##        'Positive' Class : positive
##
```

Below are attempts to upsample and downsample as discussed earlier. We did not see performance improvements, as expected.

```
downCtrl <- trainControl(
  method = "boot",
  number = 5,
  savePredictions = "final",
  classProbs = T,
  summaryFunction = twoClassSummary,
  sampling = "down"
)
upCtrl <- trainControl(
  method = "boot",
  number = 5,
  savePredictions = "final",
```

```
  classProbs = T,
  summaryFunction = twoClassSummary,
  sampling = "up"
)


rfDown <- train(target ~., data = trainingSet, method = "rf", metric = "ROC", trControl = downCtrl, tune
fittedDown <- predict(rfDown, testSet)
confusionMatrix(reference = testSet$target, data = fittedDown, mode = "everything", positive = "positive
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction negative positive
##   negative       32        7
##   positive        9       42
##
##                Accuracy : 0.8222
##                  95% CI : (0.7274, 0.8948)
##     No Information Rate : 0.5444
##     P-Value [Acc > NIR] : 2.84e-08
##
##                   Kappa : 0.6402
##
##  Mcnemar's Test P-Value : 0.8026
##
##             Sensitivity : 0.8571
##             Specificity : 0.7805
##          Pos Pred Value : 0.8235
##          Neg Pred Value : 0.8205
##               Precision : 0.8235
##                  Recall : 0.8571
##                      F1 : 0.8400
##              Prevalence : 0.5444
##          Detection Rate : 0.4667
##    Detection Prevalence : 0.5667
##       Balanced Accuracy : 0.8188
##
##        'Positive' Class : positive
##
```

```
rfUp <- train(target ~., data = trainingSet, method = "rf", metric = "ROC", trControl = upCtrl, tuneLeng
fittedUp <- predict(rfUp, testSet)
confusionMatrix(reference = testSet$target, data = fittedUp, mode = "everything", positive = "positive"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction negative positive
##   negative       33        5
##   positive        8       44
##
##                Accuracy : 0.8556
##                  95% CI : (0.7657, 0.9208)
```

```
##     No Information Rate : 0.5444
##     P-Value [Acc > NIR] : 3.463e-10
##
##                   Kappa : 0.7071
##
##  Mcnemar's Test P-Value : 0.5791
##
##             Sensitivity : 0.8980
##             Specificity : 0.8049
##          Pos Pred Value : 0.8462
##          Neg Pred Value : 0.8684
##               Precision : 0.8462
##                  Recall : 0.8980
##                      F1 : 0.8713
##              Prevalence : 0.5444
##          Detection Rate : 0.4889
##    Detection Prevalence : 0.5778
##       Balanced Accuracy : 0.8514
##
##        'Positive' Class : positive
##
```

K-Nearest Neighbors

This algorithm works by looking at a patient record and finding a k number of other records in the closest vicinity. For $k = 5$, the algorithm would look for the 5 nearest data points in relation to the patient it is trying to predict. If a majority of the patients in that search had heart disease, the model would predict the new patient to have heart disease.

```
knn <- caret::train(target ~., data = trainingSet, method = "knn")
knn
```

```
## k-Nearest Neighbors
##
## 213 samples
##  30 predictor
##   2 classes: 'negative', 'positive'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 213, 213, 213, 213, 213, 213, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.6226267  0.2390073
##   7  0.6277897  0.2484407
##   9  0.6369424  0.2667663
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```
knnGrid <- data.frame(k = c(2, 5, 7, 9, 10, 11, 12, 13, 15, 17, 19))

knnTG <- train(target ~., data = trainingSet, method = "knn", metric = "ROC", trControl = twoClassCtrl,

fittedTG <- predict(knnTG, testSet)
```

```
confusionMatrix(reference = testSet$target, data = fittedTG, mode = "everything", positive = "positive")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction negative positive
##   negative       26       18
##   positive       15       31
##
##                Accuracy : 0.6333
##                  95% CI : (0.5251, 0.7325)
##     No Information Rate : 0.5444
##     P-Value [Acc > NIR] : 0.0554
##
##                   Kappa : 0.2652
##
##  Mcnemar's Test P-Value : 0.7277
##
##             Sensitivity : 0.6327
##             Specificity : 0.6341
##          Pos Pred Value : 0.6739
##          Neg Pred Value : 0.5909
##               Precision : 0.6739
##                  Recall : 0.6327
##                      F1 : 0.6526
##              Prevalence : 0.5444
##          Detection Rate : 0.3444
##    Detection Prevalence : 0.5111
##       Balanced Accuracy : 0.6334
##
##        'Positive' Class : positive
##
```

Unfortunately, this algorithm did not perform as well as random forest, even when testing for a wide range of neighbors to look for. We would keep using the random forest model for predictions.

Logistic Regression

Logistic regression produces probabilities that are then used to classify a patient. Though it is similar to linear regression, logistic regression fits a non linear relationship using a logistic function rather than the common $y = \beta_0 + \beta_1 X$. This allows us to use this statistical technique as a classification solution rather than a regression solution.

```
heart_lreg <- glm(data = trainingSet, family = binomial, formula = target ~.)

summary(heart_lreg)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial, data = trainingSet)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1872  -0.2487   0.0557   0.3301   2.5223
##
```

```
## Coefficients: (8 not defined because of singularities)
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.295e-02  2.088e+01   0.002 0.998359
## age          3.859e-02  3.535e-02   1.092 0.275043
## sex.female   1.739e+00  8.155e-01   2.132 0.033017 *
## sex.male            NA         NA      NA       NA
## cp.0        -4.106e+00  1.202e+00  -3.416 0.000637 ***
## cp.1        -2.786e+00  1.283e+00  -2.172 0.029829 *
## cp.2        -1.512e+00  1.108e+00  -1.364 0.172542
## cp.3                NA         NA      NA       NA
## trestbps    -2.432e-02  1.483e-02  -1.640 0.101005
## chol        -1.551e-03  6.404e-03  -0.242 0.808655
## fbs.1        5.424e-01  8.199e-01   0.662 0.508282
## fbs.0               NA         NA      NA       NA
## restecg.0    1.375e+01  1.248e+03   0.011 0.991207
## restecg.1    1.371e+01  1.248e+03   0.011 0.991236
## restecg.2           NA         NA      NA       NA
## thalach      1.611e-02  1.813e-02   0.889 0.374232
## exang.0      3.162e-01  6.075e-01   0.520 0.602718
## exang.1             NA         NA      NA       NA
## oldpeak     -6.099e-01  3.305e-01  -1.845 0.064968 .
## slope.0     -1.681e+00  1.402e+00  -1.199 0.230688
## slope.1     -2.383e+00  7.231e-01  -3.296 0.000980 ***
## slope.2             NA         NA      NA       NA
## ca.0         3.847e-01  3.448e+00   0.112 0.911151
## ca.2        -2.913e+00  3.654e+00  -0.797 0.425445
## ca.1        -2.472e+00  3.485e+00  -0.709 0.478188
## ca.3        -2.120e-01  3.600e+00  -0.059 0.953042
## ca.4                NA         NA      NA       NA
## thal.0      -1.403e+01  2.400e+03  -0.006 0.995334
## thal.1       2.611e+00  1.071e+00   2.439 0.014734 *
## thal.2       2.075e+00  6.730e-01   3.084 0.002046 **
## thal.3              NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 293.58  on 212  degrees of freedom
## Residual deviance: 110.40  on 190  degrees of freedom
## AIC: 156.4
##
## Number of Fisher Scoring iterations: 15
```

```
lreg_predicts <- predict(heart_lreg, testSet, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
head(lreg_predicts)
```

```
##          1          2          3          4          5          6
## 0.70880098 0.97802916 0.50239748 0.89099838 0.98077648 0.08953616
```

```
lreg_predicts <- ifelse(lreg_predicts >= 0.5, 1, 0)
head(lreg_predicts)
```

```
## 1 2 3 4 5 6
## 1 1 1 1 1 0
```

```
lreg_confusion <- table(testSet$target, lreg_predicts)
lreg_confusion
```

```
##           lreg_predicts
##             0  1
##   negative 32  9
##   positive  5 44
```

```
sum(diag(lreg_confusion))/nrow(testSet)
```

```
## [1] 0.8444444
```

Our accuracy score on the test set was 84.4% percent, lower than our random forest. Other probability thresholds were attempted to improve accuracy, but they performed worse than out .5 threshold.

Limitations

While machine learning algorithms can be effective tools for predictions and pattern detection, they are not perfect and understanding their weaknesses can help you make more educated decisions. I think our accuracy would have been better if we had more patient records and other variables to consider. Further, the model's predictive power is unknown because we used the model to identify if a patient already had heart disease or not. While this can be useful in supplementing a physicians diagnosing process, there is untested applicability for a preventative use. Further study would be necessary to see if tweaks can be made to see what pre-heart disease numbers would look like.
