# Heart Analysis Redone

Albert Wiryawan (avw2@illinois.edu)

12/9/2020

## Table of Contents

## Change log

To improve my analysis I prevented myself from being 'lazy' and simply omitting the data that had NA values. While we did omit these values I accommodated for this by changing the problem to a binary classification problem that would be able to decide an individual as having or not having heart disease with better accuracy.

---

## Abstract

Heart disease is an illness that has a large impact on many citizens in the US as well as globally. However, prevention and reduction treatments can be employed to minimize the damage at which this disease can cause. As such, this data exploration will focus on determining whether or not someone has heart disease or not. This creates a binary classification problem in which (1) indicates someone with heart disease while (0) indicates someone without heart disease. Utilizing a five-cross fold validation, the best type of learning method was able to be determined from decision tree, knn, and random forest. The obtained metrics for accuracy were 66.66%, 78.75%, 85% for knn, decision tree, and random forest respectively. As such, the statistical learning method was tested on the created test set and obtained an overall accuracy of 77.967%.

---

## Introduction

According to a census collected by the Center for Disease Control and Prevention in 2015, heart disease has been determined as a leading cause of death deriving to 23.4% of the total deaths in the United States. Luckily, there are some habitual actions that can help prevent and control heart disease. In this data exploration, a tool will be created to screen for heart disease – essentially classifying a person as having one of five different blood types: v0 being 0 majors vessels with greater than 50% diameter narrowing (no heart disease) and v1-v4 which is having 1-4 major vessels with greater than 50% diameter narrowing. The groups having at least 1 major vessel with greater than 50% diameter narrowing will be clumped into a single group that is considered to have heart disease.

---

## Methods

Consolidating the four different types of heat disease into one will allow for the omission of data observations containing NA to not have as large of an effect on the ability to create a model. Thus, 1 is used to indicate heart disease while 0 is used to indicate no heart disease. From there the steps to train and test models using five fold cross validation is employed in order to compare the three different learning methods and see the best model accuracy that can be achieved.

### Data

In order to create and test models for the predictive tool, the overall data set is split into train and test data sets in which 80% will be used to train models while 20% is used to test them.

```
set.seed(42)
trn_idx = createDataPartition(hd$num, p = 0.80, list= TRUE)
hd_trn = hd[trn_idx$Resample1, ]
hd_tst = hd[-trn_idx$Resample1,]
```

From here the observations that are missing values will be omitted. This loss of data is will be accommodated by the consolidation of a multi-classification problem to a simpler binary-classification problem. The code that performed this consolidation can be found in the appendix below.

### Modeling

A five fold cross validation will be used over the data set that is missing no data (having no NAs) to train and test the three different model creation methods. A machine learning pipeline is employed to reduce the amount of code necessary to test different model hyper parameters and acquire prediction accuracy metrics. The code that was used to create this pipeline and test the models can be seen in the appendix below.ßß

---

## Results

The highest percent accuracy that knn was able to achieve was 66.66%, for the decision tree statistical learning model 78.75% was the highest, and finally the highest accuracy achieved by the random forest model was 85%. This indicates that the random forest learning model was the most effective in the binary classification of having heart disease or not. Taking the created random forest model and testing it on the test data set that was created, it was found that the model had a 22.033% error which indicates an overall 77.967% accuracy in the model.

---

## Discussion

While the obtained accuracy is significantly improved from that found in the first analysis, this is still an accuracy that is not effective in a medical application. As a result, this tool would be an effective indicator in flagging potential patients who would likely have a heart disease for potential awareness of a heart disease issue, but should never be used as a final decision maker. In the context of medicine, both types of errors in correctly and incorrectly diagnosing can lead to harmful implications so this must be kept in mind in understanding that this model can never act as a final indicator. Based on the last analysis, however, the culmination of the heart disease types will to create a simple binary classification model drastically improved results.

---

## Appendix

```
hd_trn_no_missing = na.omit(hd_trn)
hd_tst_no_missing = na.omit(hd_tst)

set.seed(42)
est_idx = createDataPartition(hd_trn_no_missing$num, p= 0.8, list =
TRUE)
hd_est = hd_trn_no_missing[est_idx$Resample1, ]
hd_val = hd_trn_no_missing[-est_idx$Resample1, ]

est_idx1 = createDataPartition(hd_tst_no_missing$num, p= 0.8, list =
TRUE)
hd_est1 = hd_tst_no_missing[est_idx1$Resample1, ]
hd_val1 = hd_tst_no_missing[-est_idx1$Resample1, ]

#has heart disease
hd_trn_no_missing$num[hd_trn_no_missing$num == 'v1'] = 1
hd_trn_no_missing$num[hd_trn_no_missing$num == 'v2'] = 1
hd_trn_no_missing$num[hd_trn_no_missing$num == 'v3'] = 1
hd_trn_no_missing$num[hd_trn_no_missing$num == 'v4'] = 1
```

~~~
hd_tst_no_missing$num[hd_tst_no_missing$num == 'v1'] = 1
hd_tst_no_missing$num[hd_tst_no_missing$num == 'v2'] = 1
hd_tst_no_missing$num[hd_tst_no_missing$num == 'v3'] = 1
hd_tst_no_missing$num[hd_tst_no_missing$num == 'v4'] = 1


# does not have heart disease
hd_trn_no_missing$num[hd_trn_no_missing$num =='v0'] = 0
hd_tst_no_missing$num[hd_tst_no_missing$num =='v0'] = 0


#create factors for the class variables in the data set
hd_trn_no_missing$num = factor(hd_trn_no_missing$num)
hd_trn_no_missing$location = factor(hd_trn_no_missing$location)
hd_trn_no_missing$cp = factor(hd_trn_no_missing$cp)
hd_trn_no_missing$sex = factor(hd_trn_no_missing$sex)
hd_trn_no_missing$fbs = factor(hd_trn_no_missing$fbs)
hd_trn_no_missing$restecg = factor(hd_trn_no_missing$restecg)
hd_trn_no_missing$exang = factor(hd_trn_no_missing$exang)

hd_tst_no_missing$num = factor(hd_tst_no_missing$num)
hd_tst_no_missing$location = factor(hd_tst_no_missing$location)
hd_tst_no_missing$cp = factor(hd_tst_no_missing$cp)
hd_tst_no_missing$sex = factor(hd_tst_no_missing$sex)
hd_tst_no_missing$fbs = factor(hd_tst_no_missing$fbs)
hd_tst_no_missing$restecg = factor(hd_tst_no_missing$restecg)
hd_tst_no_missing$exang = factor(hd_tst_no_missing$exang)
```


### Modeling
A five fold cross validation will be used over the data set that is
missing no data (having no NAs) to train and test the three different
model creation methods. A machine learning pipeline is employed to
reduce the amount of code necessary to test different model hyper
parameters and acquire prediction accuracy metrics.
```{r, echo= FALSE, warning= FALSE}
#create a 5-fold cross-validation
cv_5= trainControl(method = "cv" , number =  5)

hd_tree_tune = expand.grid(
  cp = c(0, 0.0001, 0.001, 0.01, 0.1, 1)
)

hd_knn_tune = expand.grid(
  k = 1:100
~~~

```
)


hd_tree_mod = train(
  form = num ~.,
  data = hd_trn_no_missing,
  method = "rpart",
  trControl = cv_5,
  tuneLength = 10
)

hd_knn_mod = train(
  form = num ~.,
  data = hd_trn_no_missing,
  method = "knn",
  trControl = cv_5,
  tuneLength = 100
)


hd_rf_mod = train(
  form = num ~.,
  data = hd_trn_no_missing,
  method= "rf",
  trControl = cv_5,
  verbose = FALSE
)

hd_tree_mod
hd_knn_mod
hd_rf_mod

tst_pred_un1 = predict(hd_rf_mod, hd_tst_no_missing, type = "raw")


calc_class_err = function(actual, predicted) {
  mean(actual != predicted)
}

error1 = calc_class_err(hd_tst_no_missing$num, tst_pred_un1)
```